

Voronoi diagrams on planar graphs, and computing the diameter in deterministic $\tilde{O}(n^{5/3})$ time

Paweł Gawrychowski* Haim Kaplan† Shay Mozes‡ Micha Sharir§
Oren Weimann¶

Abstract

We present a deterministic algorithm that computes the diameter of a directed planar graph with real arc lengths in $\tilde{O}(n^{5/3})$ time. This improves the recent breakthrough result of Cabello (SODA'17), both by improving the running time (from $\tilde{O}(n^{11/6})$), and by using a deterministic algorithm. It is in fact the first truly subquadratic deterministic algorithm for this problem.

Our algorithm follows the general high-level approach of Cabello, but differs in the way it is implemented. Our main technical contribution is an explicit and efficient construction of additively weighted Voronoi diagrams on planar graphs (under the assumption that all Voronoi sites lie on a constant number of faces). The idea of using Voronoi diagrams is also borrowed from Cabello, except that he uses abstract Voronoi diagrams (as a black box), which makes the implementation more involved, more expensive, and randomized. Our explicit construction avoids these issues, and leads to the improved (and deterministic) running time.

As in Cabello's work, our algorithm can also compute the Wiener index of (sum of all pairwise distances in) a planar graph, within the same bounds.

*University of Haifa, Department of Computer Science, gawry@mimuw.edu.pl.

†Tel Aviv University, Blavatnik School of Computer Science, haimk@post.tau.ac.il.

‡Interdisciplinary Center Herzliya, Efi Arazi School of Computer Science, smozes@idc.ac.il.

§Tel Aviv University, Blavatnik School of Computer Science, michas@post.tau.ac.il.

¶University of Haifa, Department of Computer Science, oren@cs.haifa.ac.il.

1 Introduction

Computing the diameter of a graph (the largest distance between a pair of vertices) is a fundamental problem in graph algorithms, with numerous research papers studying its complexity.

For general graphs, the current fastest way to compute the diameter is by computing all-pairs shortest paths (APSP). Computing APSP can be done in cubic $O(n^3)$ time with the classical Floyd-Warshall algorithm [28, 53]. Following a long line of improvements by polylogarithmic factors [15, 16, 23, 30–33, 51, 52, 59], the current fastest APSP algorithm is the $O(n^3/2^{\Omega(\log n)^{1/2}})$ -time algorithm by Williams [54]. However, no truly subcubic, i.e. $O(n^{3-\epsilon})$ -time algorithm is known for either APSP or DIAMETER. We also do not know if DIAMETER is as hard as APSP (i.e., if a truly subcubic algorithm for DIAMETER implies a truly subcubic algorithm for APSP). This is in contrast to many other related problems that were recently shown to be as hard as APSP [1–4, 6, 9, 10, 48, 55, 56] (including the problem of computing the radius of a graph [2]).

For sparse graphs (with $\tilde{O}(n)$ edges), the problem is better understood. We can solve APSP (and thus DIAMETER) in $\tilde{O}(n^2)$ time by running a single-source shortest path algorithm from every vertex. There is clearly no truly subquadratic $O(n^{2-\epsilon})$ algorithm for APSP (as the output size is $\Omega(n^2)$). Interestingly, assuming the strong exponential time hypothesis (SETH), there is also no truly subquadratic algorithm for DIAMETER. In fact, even distinguishing between diameter 2 and 3 requires quadratic time assuming the SETH [47]. This holds even for undirected, unweighted graphs with treewidth $O(\log n)$. For graphs of bounded treewidth, the diameter can be computed in near-linear time [5] (see also [25, 38] for algorithms parameterized by treewidth and diameter). Near-linear time algorithms were developed for many other restricted graph families, see e.g. [7, 11, 17–20, 24, 27, 37, 46].

For planar graphs, we know that the diameter can in fact be computed faster than APSP. The first indication of this was illustrated by Wulff-Nilsen who gave an algorithm to compute the diameter of unweighted, undirected planar graphs in $O(n^2 \log \log n / \log n)$ time [57] and of weighted directed planar graphs in $O(n^2 (\log \log n)^4 / \log n)$ time [58]. The question of whether a truly subquadratic ($O(n^{2-\epsilon})$) algorithm exists (even for undirected, unweighted planar graphs) was a major open problem in planar graph algorithms. In SODA’17, a breakthrough result of Cabello [13] showed that the diameter of weighted directed planar graphs can be computed in truly subquadratic $\tilde{O}(n^{11/6})$ expected time (the \tilde{O} notation hides polylogarithmic factors). Whether the exponent 11/6 could be substantially reduced was now the main open problem.

Voronoi diagrams on planar graphs. The breakthrough in Cabello’s work is his novel use of Voronoi diagrams in planar graphs.¹ Let G be a directed planar graph with non-negative arc lengths. An r -division [29] is a decomposition of G into $\Theta(n/r)$ pieces, each of them with $O(r)$ vertices and $O(\sqrt{r})$ boundary vertices (vertices shared with other pieces). There is an $O(n)$ -time algorithm that computes an r -division with the additional property that the boundary vertices in each piece lie on a constant number of faces of the piece (called *holes*) [26, 41]. Unsurprisingly, it turns out that the difficult case for computing the diameter is when the farthest pair of vertices lie in different pieces.

Consider some source vertex v_0 outside of some piece P . For every boundary vertex u of P , let $\omega(u)$ denote the v_0 -to- u distance in G . The *additively weighted Voronoi diagram* of P with respect to $\omega(\cdot)$ is a partition of the vertices of P into pairwise disjoint sets (Voronoi cells), each associated with a unique boundary vertex (site) u . The vertices in the cell $\text{Vor}(u)$ of u are all vertices v of P such that u minimizes $\omega(u)$ plus the length of the u -to- v shortest path inside P (i.e., u is the last

¹Using Voronoi diagrams for algorithms in planar graphs was done before (see e.g. [22, 45]), but never for a polynomially solvable problem (where we care about polynomial factors).

boundary vertex on the v_0 -to- v shortest path in G). For every v_0 , computing the farthest vertex from v_0 in P then boils down to computing, for each site u , the farthest vertex from u in $\text{Vor}(u)$.

The main challenges with this approach are: (1) to efficiently compute the Voronoi diagram of a piece P (that is, to identify the partition of the vertices of P into Voronoi cells) with respect to boundary weights being the v_0 -to-boundary distances, and (2) to find the maximum site-to-vertex distance in each cell. To appreciate these issues, note that we need to perform tasks (1) and (2) much faster than $O(|P|) = O(r)$ time! Otherwise, since there are n possible sources and $\Theta(n/r)$ pieces, the overall running time would be $\Omega(n^2)$. Thus, even task (2), which is trivial to implement by simply inspecting all vertices of P , requires a much more sophisticated approach.

Voronoi diagrams of geometric objects of many kinds and with respect to different metrics have been extensively studied (see, e.g., [8]). One of the basic approaches to compute Voronoi diagrams is divide and conquer. Using this approach we split the objects (sites) into two groups, compute the Voronoi diagram of each group recursively, and then somehow merge the diagrams. Using this approach, Shamos and Hoey [50] gave the first $O(n \log n)$ deterministic algorithm for computing the Voronoi diagram of n points in the plane with respect to the Euclidean metric. Since then Other approaches, such as randomized incremental construction, have also been developed.

Cabello’s approach. For every source v_0 and every piece P , Cabello uses the “heavy hammer” of *abstract Voronoi diagrams* in order to compute the Voronoi diagram of P with respect to the v_0 -to-boundary distances as additive weights. Klein [42] introduced the abstract Voronoi diagram framework, trying to abstract the properties of objects and the metric that make the computation of Voronoi diagrams efficient. Cabello uses (as a black box) the randomized construction of abstract Voronoi diagrams by Klein, Mehlhorn and Meiser [44] (see also [43]). This construction is highly efficient, requiring only $\tilde{O}(\sqrt{r})$ time. However, in order to use it, certain requirements must be met.

First, it requires knowing in advance the Voronoi diagrams induced by every subset of four sites (boundary vertices). Cabello addresses this by first showing that the planar case only requires the Voronoi diagrams for any triple of sites to be known in advance. The Voronoi diagram of each triple is composed of segments of *bisectors* – boundaries of the Voronoi diagrams of pairs of sites. Cabello shows that for each pair of sites, there are only $O(r)$ different bisectors over all possible weights for this pair. He computes each of these bisectors separately in $O(r)$ time, so computing all the bisectors for a single pair of sites takes him $O(r^2)$ time. Then, he precomputes all 3-site weighted Voronoi diagrams from the bisectors of pairs of sites in $O(r^{7/2})$ time.

The second requirement is that all the sites must lie on a single hole. However, from the properties of r -division, the boundary vertices of P may lie on many (albeit a constant number of) holes. Overcoming this is a significant technical difficulty in Cabello’s paper. He achieves this by allowing the bisectors to venture through the subgraphs enclosed by the holes (i.e., through other pieces). Informally, Cabello’s algorithm “fills up” the holes in order to apply the mechanism of abstract Voronoi diagrams.

Our result. We present a deterministic $\tilde{O}(n^{5/3})$ -time algorithm for computing the diameter of a directed planar graph with no negative-length cycles. This improves Cabello’s bound by a polynomial factor, and is the first deterministic truly subquadratic algorithm for this problem.

Our algorithm follows the general high-level approach of Cabello, but differs in the way it is implemented. Our main technical contribution is an explicit and efficient construction of additively weighted Voronoi diagrams on planar graphs (under the assumption that the sites lie on the boundaries of a constant number of faces, which holds in the context considered here). In contrast to Cabello, we only need to precompute bisectors since we developed a technique for intersecting three bisectors in $\tilde{O}(1)$ time. Moreover, we compute the bisectors faster, in $\tilde{O}(r)$ time for all bisectors for a given pair compared to $O(r^2)$ time in [13]. Cabello’s use of abstract Voronoi diagrams makes the

implementation more involved, more expensive, and randomized. Our explicit construction avoids these issues, and leads to the improved (and deterministic) running time. This is the first explicit non-trivial construction of Voronoi diagrams on planar graphs. Formally, we prove the following:

Theorem 1.1. *Let P be a directed planar graph with real arc lengths, r vertices, and a set S of b sites that lie on the boundaries of a constant number of faces (holes). One can preprocess P in $\tilde{O}(r \cdot b^2)$ time, so that given any subset $S' \subseteq S$ of the sites, and a weight $\omega(u)$ for every $u \in S'$, one can construct a representation of the additively weighted Voronoi diagram of S' with respect to the weights $\omega(\cdot)$ in $\tilde{O}(|S'|)$ time. With this representation of the Voronoi diagram we can, for any site $u \in S'$, (i) query the maximum distance from u to a vertex in the Voronoi cell of u in $\tilde{O}(|S'|)$ time, and (ii) report the boundary of the Voronoi cell of u in $\tilde{O}(1)$ time per edge.*

The above $\tilde{O}(|S'|)$ construction time is significant because $|S'| \leq b \ll |P| = O(r)$, and the dependence on r is only polylogarithmic. This fast construction comes with the price of the cost of the preprocessing stage. In other words, a one-time construction of the diagram is less efficient than a brute force approach, because we need to account for the preprocessing cost too. Nevertheless, the preprocessing is *independent* of the weights of the sites. Hence, the overall approach, which consists of a one-time preprocessing stage (per piece), followed by many calls to the diagram construction procedure, makes the overall algorithm efficient. This general approach is borrowed from Cabello, except that we achieve a more efficient and deterministic implementation. The high level view of our diameter algorithm is slightly simpler than Cabello's. Since we can efficiently construct Voronoi diagrams with sites on a constant number of holes, we can avoid the complications associated with filling up the holes. However, because of the multiple holes, we also need to enhance Cabello's mechanism for reporting the maximum site-to-vertex distance in a cell.

As in Cabello's work, our algorithm can also compute the Wiener index of (sum of all pairwise distances in) a planar graph, within the same performance bounds. We believe that our detailed study of Voronoi diagrams for planar graphs, and the resulting algorithm of Theorem 1.1 is of independent interest. For example, in a very recent work [21] it was shown that the Voronoi diagram approach can be used to construct a *distance oracle* for planar graphs of size $\tilde{O}(n^{5/3})$, query time $O(\log n)$, and preprocessing time $O(n^2)$. No oracle with subquadratic space and polylogarithmic query time was previously known. Using Theorem 1.1 reduces their preprocessing time from $O(n^2)$ to $\tilde{O}(n^{5/3})$ (while retaining their space and query time bound).

1.1 A high-level description of the diameter algorithm

Let G be a directed planar graph with non-negative arc lengths. The algorithm computes an r -division of G with few holes. For a vertex v that is not a boundary vertex, let P_v be the piece of the r -division that contains v . The diameter of G is the length of a shortest \hat{u} -to- \hat{v} path for some vertices \hat{u}, \hat{v} . There are three cases: (i) at least one of \hat{u}, \hat{v} is a boundary vertex, (ii) none of \hat{u}, \hat{v} is a boundary vertex and $P_{\hat{u}} = P_{\hat{v}}$, and (iii) none of \hat{u}, \hat{v} is a boundary vertex and $P_{\hat{u}} \neq P_{\hat{v}}$.

To take care of case (i), for each vertex $v \in G$, the algorithm computes the distances in G from v to all the boundary vertices of all pieces. This is done by first computing the distances in P_v from v to ∂P_v in $O(r)$ time using the single-source shortest paths algorithm of Henzinger et al. [36].² Next, we compute the *dense distance graph* (DDG) of every piece P in the r -division. The DDG of a piece P is the complete graph over the boundary vertices ∂P of P , where the length of edge uv in the DDG of P equals to the u -to- v distance inside P (note that the DDG of P is in general non-planar). The DDGs of all pieces can be computed in total $O(n \log r)$ time using the MSSP

²In fact, for our purposes it suffices to use Dijkstra's algorithm in $O(r \log r)$ time.

algorithm of Klein [40]. Finally, using the DDGs and the computed v -to- ∂P_v distances in P_v , we compute the distances in G from v to all the boundary vertices of all pieces in $\tilde{O}(r + n/\sqrt{r})$ time using an implementation of Dijkstra's algorithm (nicknamed *FR-Dijkstra*) due to Fakcharoenphol and Rao [26]. This takes care of case (i). To take care of case (ii), the algorithm next computes the distances in G from v to all vertices of P_v . This is done by a single-source shortest-path algorithm on P_v , with the distance label of vertices of ∂P_v initialized to their distance from v in the full graph G (these distances were already computed in case (i)).

It remains to take care of case (iii), that is to compute the maximum distance between vertices that are not in the same piece. For every non-boundary source vertex $v_0 \in G$, and every piece P in the r -division (except for the piece of v_0) we compute, in $\tilde{O}(\sqrt{r})$ time, the farthest vertex from v_0 in P . We achieve this using the weighted Voronoi diagram of P , where the weight $\omega(v)$, for each $v \in \partial P$ is the v_0 -to- v distance in G (these distances have been computed in case (i)). In Section 3 we will describe how to preprocess P in $\tilde{O}(r^2)$ time to compute a representation of all weighted bisectors for all pairs of vertices $u, v \in \partial P$ and for all possible values of $w(v)$ and $w(u)$. Given this representation, we present, in Section 6 an algorithm that computes the weighted Voronoi diagram of P with respect to any given weight assignment, in $\tilde{O}(|\partial P|) = \tilde{O}(\sqrt{r})$ time. Finally, given this weighted Voronoi diagram of P , we describe in Section 7 an algorithm that, given a site v in ∂P , returns the farthest vertex w from v_0 in the Voronoi cell of site v . This can be done in time nearly linear in the number of Voronoi vertices in v 's cell. (a Voronoi vertex of the cell of v is the dual of a face of P that has vertices in three different Voronoi cells, one of them being the cell of v). We then apply this algorithm to all sites $v \in \partial P$. The total number of Voronoi vertices over all cells in the Voronoi diagram of P is $O(|\partial P|) = O(\sqrt{r})$. Therefore, the time to report the maximum distance from v_0 to all vertices in P is $\tilde{O}(\sqrt{r})$, as claimed. The total preprocessing time over all pieces is $\tilde{O}(\frac{n}{r} \cdot r^2) = \tilde{O}(nr)$, and the total query time over all sources v_0 and all pieces P is $\tilde{O}(n \cdot \frac{n}{r} \cdot \sqrt{r}) = \tilde{O}(n^2/\sqrt{r})$. Summing the preprocessing and query bounds, we get an overall cost of $\tilde{O}(nr + n^2/\sqrt{r})$.

Concerning the running time of the entire algorithm, we note that computing the r -division and the DDGs takes $O(n \log r)$ time [40, 41]. Each invocation of the FR-Dijkstra algorithm [26] requires $\tilde{O}(r + n/\sqrt{r})$ time, for a total of $\tilde{O}(nr + n^2/\sqrt{r})$ time. Single-source shortest paths inside a piece are computed in $O(r)$ time [36], contributing another $O(nr)$ term to the running time. The total running time is thus $\tilde{O}(nr + n^2/\sqrt{r})$. Setting $r = n^{2/3}$ yields the claimed $\tilde{O}(n^{5/3})$ bound.

1.2 A high level description of the Voronoi diagrams algorithm

We now outline the proof of Theorem 1.1. The description is given in the context of the diameter algorithm, so the input planar graph is called a piece P , and the faces to which the sites are incident are called holes. The set of sites is denoted by S . In this description we assume that the subset S' of sites of the desired Voronoi diagram is the entire set S (since this is the case in the diameter application). Hence, $|S'| = |S| = b = O(\sqrt{r})$. Given a weight assignment $\omega(\cdot)$ to the sites in S , the algorithm computes (an implicit representation of) the *additively weighted Voronoi diagram* of S inside P , denoted as $VD(S, \omega)$, or just $VD(S)$ for short. Recall that our goal is to construct many instances of $VD(S)$ in $\tilde{O}(b) = \tilde{O}(\sqrt{r})$ time each. What enables us to achieve this is the fact that, these instances differ only in the weight assignment $\omega(\cdot)$. We exploit this by carrying out a preprocessing stage, which is weight-independent, and use the information collected there to make the construction of the diagrams efficient as desired.

The structure of $VD(S)$. Each Voronoi cell $\text{Vor}(v)$ is in fact a tree rooted at v , which is a subtree of the shortest-path tree from v (within P). We also consider the dual graph P^* of P , and

use the following dual representation of $VD(S)$, which we denote as $VD^*(S)$, within P^* . For each pair of distinct sites u, v , we define the *bisector* $\beta^*(u, v)$ of u and v to be the collection of all arcs $(pq)^*$ of P^* that are dual to arcs pq of P for which p is nearer to u than to v and q is nearer to v than to u . (As already noted, $\beta^*(u, v)$ constitutes the Voronoi diagram of just the two sites u, v .) Each bisector is a simple cycle in P^* . The set of maximal segments of $\beta^*(u, v)$ that consist of arcs $(pq)^*$ for which $p \in \text{Vor}(u)$ and $q \in \text{Vor}(v)$, is a collection of one or several *Voronoi edges* that separate between the cells $\text{Vor}(u)$ and $\text{Vor}(v)$. Each Voronoi edge terminates at a pair of *Voronoi vertices*, where such a vertex f^* is dual to a face f of P whose vertices are distributed among three or more distinct Voronoi cells. To simplify matters, we triangulate each face of P (except for the holes), so all Voronoi vertices (except those dual to the holes, if any) are of degree 3. If we contract each Voronoi edge into the single edge connecting its endpoints, we get a planar map of size $O(b)$ (by Euler’s formula), which is the dual Voronoi diagram $VD^*(S)$. An additional useful property is that, in the presence of just three sites, the diagram has at most two vertices. In fact, assuming that the sites lie on the boundaries of only three holes, the diagram has at most two “trichromatic” vertices, that is, vertices whose dual faces have vertices in the Voronoi cells of sites from each hole. See Sections 2 and 5.

Computing all bisectors. The heart of the preprocessing step is the computation of bisectors of every pair of sites u, v , and every possible pair of weights $\omega(u), \omega(v)$ that can be assigned to them. Note that the $\beta^*(u, v)$ only depends on the *difference* $\delta = \omega(v) - \omega(u)$ between the weights of these sites, and that, due to the discrete nature of the setup, the bisector changes only at a discrete set of differences, so there are $O(r)$ different bisectors for each pair (u, v) . Computing a representation of the $O(r)$ possible bisectors for a fixed pair of sites (u, v) is done in $\tilde{O}(r)$ time by varying δ from $+\infty$ to $-\infty$. As we do this, the bisector “sweeps” over the piece, moving farther from u and closer to v . This is reminiscent of the multiple source shortest paths (MSSP) algorithm of [14, 40], except that in our case u and v do not necessarily lie on the same face. We represent all the possible bisectors for (u, v) using persistent binary search trees [49] in $\tilde{O}(r)$ space. Summing over all $O((\sqrt{r})^2) = O(r)$ pairs of sites, the total preprocessing time is $O(r^2)$. See Section 3.

Computing $VD(S)$. We compute the diagram in four steps: (i) We first compute the diagram for the sites on the boundary of a single hole (the “monochromatic” case). We do this using a divide-and-conquer technique, whose main ingredient is merging two sub-diagrams into a larger diagram. We obtain t different diagrams, one for each hole. (ii) We then compute the diagram of the sites that lie on the boundaries of a fixed pair of holes (the “bichromatic case”), for all $\binom{t}{2} = O(1)$ pairs of holes. Each such diagram is obtained by merging the two corresponding monochromatic diagrams. (iii) We then compute all “trichromatic” diagrams, each of which involves the sites that lie on the boundaries of three specific holes. Here too we merge the three (already computed) corresponding bichromatic diagrams. (iv) Since each Voronoi edge (resp., vertex) is defined by two (resp., three) sites, we now have a superset of the vertices of $VD(S)$, and each Voronoi edge of $VD(S)$, say separating the cells of sites u, v , is contained in the appropriate Voronoi edges of all trichromatic diagrams involving the holes of u and v . A simple merging step along each bisector constructs the true Voronoi edges and vertices. See section 6.

Intersecting bisectors, finding trichromatic vertices, and merging diagrams. The main technical part of the algorithm is an efficient implementation of steps (i)–(iii) above. A crucial procedure that the algorithm repeatedly uses is an efficient construction of the (at most two) Voronoi vertices of three sites, where the cost of this step is only $\tilde{O}(1)$. In fact, we generalize this procedure so that it can compute the (at most two) trichromatic Voronoi vertices, of a diagram of the form $VD(\{r, g\} \cup B)$, where r and g are two sites and B is a set of sites on a fixed hole. This extended version also takes only $\tilde{O}(1)$ time.

Having such a procedure at hand, each merging of two subdiagrams $VD(S_1)$, $VD(S_2)$, into the joint diagram $VD(S_1 \cup S_2)$, is performed by tracing the $S_1 S_2$ -bisector, which is the collection of all Voronoi edges of the merged diagram that separate between a cell of a site in S_1 and a cell of a site in S_2 . In all the scenarios where such a merge is performed, the $S_1 S_2$ -bisector is also a cycle, which we can trace segment by segment. In each step of the trace, we are on a bisector of the form $\beta^*(u_1, u_2)$, for $u_1 \in S_1$ and $u_2 \in S_2$. We then take the cell of u_1 in $VD(S_1)$, and compute the trichromatic vertices for u_1 , u_2 , and $S_1 \setminus \{u_1\}$. We do the same for the cell of u_2 in $VD(S_2)$, and the two steps together determine the terminal vertex of the portion of $\beta^*(u_1, u_2)$ that we trace, in the combined diagram. See Section 6.

The details concerning the construction of trichromatic vertices are rather involved, and we only give a few hints in this overview. Let r, g, b be our three sites. We keep the bisector $\beta^*(g, b)$ fixed, meaning that we keep the weights $\omega(g)$, $\omega(b)$ fixed, and vary the weight $\omega(r)$ from $+\infty$ to $-\infty$. As already reviewed earlier, this causes the cell $\text{Vor}(r)$, which is initially empty, to gradually expand, “sweeping” through P . This expansion occurs at discrete critical values of $\omega(r)$. We show that, as $\text{Vor}(r)$ expands, it annexes a contiguous portion of $\beta^*(g, b)$, which keeps growing as $\omega(r)$ decreases. The terminal vertices of the annexed portion of $\beta^*(g, b)$ are the desired trichromatic vertices. By a rather involved variant of binary search through $\beta^*(g, b)$ (with the given $\omega(r)$ as a key), we find those endpoints, in $\tilde{O}(1)$ time. See Section 4.

To support this binary search we need to store, at the preprocessing step, for each vertex p of P and for each pair of sites r, g , the “time” at which $\beta^*(r, g)$ sweeps through p . Note that this can be done within the $O(r^2)$ time and space that it takes to precompute the bisectors.

Putting all the above ingredients together (where the details that we skimmed through in this overview are spelled out later on), we get an overall algorithm that constructs the Voronoi diagram, within a single piece and under a given weight assignment, in $\tilde{O}(b) = \tilde{O}(\sqrt{r})$ time.

Finding the farthest vertex in each Voronoi cell. To be useful for the diameter algorithm, our representation of Voronoi diagrams must be augmented to report the farthest vertex in each Voronoi cell in nearly linear time in the number of Voronoi vertices of the cell. Such a mechanism has been developed by Cabello, but only for pieces with a single hole. We extend this procedure to pieces with a constant number of holes by exploiting the structure of Voronoi diagrams, and the interaction between a shortest path tree, its cotree, and the Voronoi diagram. See Section 7.

2 Preliminaries

Planar embedded graphs. We assume basic familiarity with planar embedded graphs and planar duality. We treat the graph $G = (V, E)$ as a directed object, where V is the set of vertices, and E is the set of arcs. We assume that for every arc $e = uv$ there is an anti-parallel arc $\text{rev}(e) = vu$ that is embedded on the same curve in the plane as e . We call u the *tail* of e and v the *head* of v . We use the term *edge* when the direction plays no role (i.e., when we wish to refer to the undirected object, not distinguishing between the two antiparallel arcs). The dual of a planar graph G is a planar embedded graph $G^* = (V^*, E^*)$. The nodes in V^* represent faces in G , and the dual arcs in E^* stand in 1-1 correspondence with the arcs in E , in the sense that the arc e^* dual to an arc e connects the face to the left of e to the face to the right of e .

We use some well known properties of planar graphs. See, e.g., [39]. If T is a spanning tree of G then the edges not in T form a spanning tree T^* of the dual G^* . The tree T^* is called the *co-tree* of T . If (S, T) is a partition of V , and the subgraphs induced by S and by T are both connected, then the set of duals of the arcs whose tail is in S and whose head is in T form a simple cycle in G^* .

Assumptions. By a *piece* $P = (V, E)$ we mean an embedded directed planar graph with $t = O(1)$ distinguished faces h_1, \dots, h_t , to which we refer as *holes*. (In our context, the pieces are the subgraphs of the input graph G produced by an r -division.) Each arc $uv \in E$ has a *length* $\ell(uv)$ associated with it. We assume, for the diameter algorithm, that arc lengths are non-negative. This assumption can be enforced using the standard technique of price functions and reduced lengths (see e.g. [13, 26]). For the Voronoi diagrams algorithm we only assume that there are no negative cycles so shortest paths are well defined. We assume that all faces of G , except possibly for the holes, are triangulated. (When we triangulate a non-triangular face by diagonals, we assign length $= +\infty$ to the arcs associated with the diagonal.) Except for the nodes representing the holes, all other nodes of P^* have therefore degree 3.

We assume that shortest paths are unique. Our specific assumption is identical to the one made in [14]. Let P be a piece with a set S of sites and additive weights $\omega(\cdot)$. Consider the process in which we vary the weight $\omega(u)$ of exactly one site $u \in S$. We assume that vertex-to-vertex distances in P are distinct, and that shortest paths are unique, except at critical values of $\omega(u)$ where there is a unique *tense* arc (see Section 3). This assumption can be achieved deterministically with $\tilde{O}(1)$ time overhead time using lexicographic comparison [14, 34].

Weighted Voronoi diagram in a piece. We are given a piece $P = (V, E)$ with $|V| = O(r)$ (and so $|E| = O(r)$). We are also given a set $S \subseteq V$ of $b = O(\sqrt{r})$ vertices, each of which is a vertex of one of the holes h_1, \dots, h_t ; S is a subset of the boundary vertices of the piece P , and its elements are the *sites* of the Voronoi diagram that we are going to define. Each site $v \in S$ has a *weight* $\omega(v) \geq 0$ associated with it. We think of ω as a weight function on S . For every pair of vertices v and w , we denote by $\pi(u, v)$ the length of the shortest path from u to v in P . The *distance* between a site $u \in S$ and a vertex $v \in V$, denoted by $d(u, v)$, is $\omega(u)$ plus the length of $\pi(u, v)$.

The *additively weighted Voronoi diagram* of (S, ω) within P , denoted by $VD(S, \omega)$ (we will often drop ω from this notation, although the diagram does depend on ω), is a partition of V into pairwise disjoint sets, one set, denoted by $\text{Vor}(u)$, for each site $u \in S$. The set $\text{Vor}(u)$ contains all vertices closer to u than to any other site $u' \neq u \in S$. We call $\text{Vor}(u)$ the (primal) *Voronoi cell* of u . Note that a Voronoi cell might be empty. The Voronoi diagram has the following basic property. In what follows, we denote the shortest-path tree rooted at a site $u \in S$ as T_u . For brevity, we defer the proofs of the following lemmas to Section 5.

Lemma 2.1. *For each $u \in S$, the vertices in $\text{Vor}(u)$ form a connected subtree (rooted at u) of T_u .*

Let $B \subseteq E$ be the set of edges $vw \in E$ such that the sites u_1, u_2 , for which $v \in \text{Vor}(u_1)$ and $w \in \text{Vor}(u_2)$ are distinct. Let B^* denote the set of the edges that are dual to the edges of B , and let $VD^*(S, \omega)$ (or $VD^*(S)$ for short) denote the subgraph of P^* with B^* as a set of edges.

The following consequence of Lemma 2.1 gives some structural properties of $VD^*(S)$.

Lemma 2.2. *The graph $VD^*(S)$ consists of at most b faces, so that each of its faces corresponds to a site $u \in S$ and is the union of all faces of P^* that are dual to the vertices of $\text{Vor}(u)$.*

We refer to the face of $VD^*(S)$ corresponding to a site $u \in S$ as the *dual Voronoi cell* of u , and denote it as $\text{Vor}^*(u)$. ($\text{Vor}^*(u)$ is empty when $\text{Vor}(u)$ is) By Lemma 2.2, we can think of $\text{Vor}^*(u)$ as the union of the faces dual to the vertices of $\text{Vor}(u)$. Once we fix a concrete way in which we draw the dual edges in B^* in the plane, we can regard each $\text{Vor}^*(u)$ as a concrete embedded planar region. Since the sets $\text{Vor}(u)$ form a partition of V , it follows that the sets $\text{Vor}^*(u)$ induce a partition of the sets of dual faces of P^* .

We define a vertex $f^* \in VD^*(S)$ to be a *Voronoi vertex* if its degree in $VD^*(S)$ is at least 3. This means that there exist at least three distinct sites whose primal Voronoi cells contain vertices

of the primal face f dual to f^* . The next corollary follows directly from Euler's formula for planar graphs.

Corollary 2.1. *The graph $VD^*(S)$ consists of $O(b)$ vertices of degree ≥ 3 ; all other vertices are of degree 2. The only vertices of degree strictly larger than 3 are those corresponding to the non-triangular holes among h_1, \dots, h_t .*

Lemma 2.3. *For any three distinct sites u, v, w in S there are at most two faces f of P such that each of the cells $\text{Vor}(u)$, $\text{Vor}(v)$, $\text{Vor}(w)$ contains a vertex of f .*

Let u and v be two distinct sites in S . The *bisector* between u and v (with respect to their assigned weights), denoted as $\beta^*(u, v)$, is defined to be the set of arcs of P^* whose corresponding primal edges have their tail in $\text{Vor}(u)$ and their head in $\text{Vor}(v)$. Note that, unless we explicitly say otherwise, the bisector $\beta^*(u, v)$ is a directed object, and $\beta^*(v, u)$ consists of the reverses of the arcs of $\beta^*(u, v)$. Viewed as an undirected object, $\beta^*(u, v)$ is the dual Voronoi diagram of the doubleton set $\{u, v\}$. Bisectors satisfy the following crucial property.

Lemma 2.4. *$\beta^*(u, v)$ is a simple cycle of arcs of P^* . If u and v are incident to the same hole h and $\beta^*(u, v)$ is nonempty then $\beta^*(u, v)$ is incident to h^* .*

3 Computing the bisectors during preprocessing

To facilitate an efficient implementation of the algorithm, we carry out a preprocessing stage, in which we compute the bisectors $\beta^*(u, v)$ for every pair of sites $u, v \in S$ and for every pair of weights that can be assigned to u and v . To clarify this statement, note that $\beta^*(u, v)$ only depends on the *difference* $\delta = \omega(v) - \omega(u)$ between the weights of u and v , and that, due to the discrete nature of the setup, $\beta^*(u, v)$ changes only at a discrete set of differences. The preprocessing stage computes, for each pair $u, v \in S$, all possible bisectors, by varying δ from $+\infty$ to $-\infty$. As we do this, $\beta^*(u, v)$ “sweeps” over P^* , moving farther from u and closer to v , in a sense that will be made more precise shortly. We find all the critical values of δ at which $\beta^*(u, v)$ changes, and store all versions of $\beta^*(u, v)$ in a (partially) persistent binary search tree [49]. Each version of the bisector is represented as a binary search tree on the (cyclic) list of its dual vertices and edges (which we cut open at some arbitrary point). Hence, we can find the k -th edge on any bisector $\beta^*(u, v)$ in $O(\log |\beta^*(u, v)|) = O(\log r)$ time, for any $1 \leq k \leq |\beta^*(u, v)|$ (where $|\beta^*(u, v)|$ denotes the number of edges on the bisector $\beta^*(u, v)$).

Consider a critical value of $\delta = \omega(v) - \omega(u)$ at which $\beta^*(u, v)$ changes. We assume (see Section 2) that there is a unique arc $yz \in T_v$ such that for $\delta' > \delta$, $y \in \text{Vor}(v)$ and $z \in \text{Vor}(u)$, and $\delta = \pi(u, z) - \pi(v, z)$. We say that yz is *tense* at δ . Just before reaching δ , z was a node in the shortest-path (sub)tree $\text{Vor}(u)$, and right afterwards it becomes a node of $\text{Vor}(v)$. If z was not a leaf of $\text{Vor}(u)$ (at the time of the switch), then the entire subtree rooted at z moves with z from $\text{Vor}(u)$ to $\text{Vor}(v)$ (this is an easy consequence of the property that shortest paths do not meet or cross one another).

The sweep crucially depends on the following lemma, whose proof is deferred to Section 5.

Lemma 3.1. *Consider some critical value δ . The dual edges that newly join $\beta^*(u, v)$ at δ form a contiguous portion of the new bisector, and the dual edges that leave $\beta^*(u, v)$ at δ form a contiguous portion of the old bisector.*

We compute and store, for each vertex p , and for each pair of sites u, v , the unique value of δ at which it moves from $\text{Vor}(u)$ to $\text{Vor}(v)$. Denote this value as $\delta^{uv}(p)$. Consider an arc pq of P . If $\delta^{uv}(p) = \delta^{uv}(q)$ (i.e. pq is not the tense arc) then both endpoints of the arc move together from $\text{Vor}(u)$ to $\text{Vor}(v)$, so the dual arc never becomes an arc of $\beta^*(u, v)$. We cannot have $\delta^{uv}(p) > \delta^{uv}(q)$ since the $\delta^{uv}()$ of any node cannot be smaller than the $\delta^{uv}()$ of its parent. However, if $\delta^{uv}(p) < \delta^{uv}(q)$ then q moves first and at time $\delta^{uv}(q)$ the edge dual to pq becomes an edge of $\beta^*(u, v)$. It stops being an edge of $\beta^*(u, v)$ at $\delta^{uv}(p)$.

We compute the bisectors $\beta^*(u, v)$ by adding and deleting arcs at the appropriate values of δ . Note that there are $O(r)$ pairs of sites in S , so for each vertex $p \in V$ we compute and store $O(r)$ values of δ , for a total of $O((n/r) \cdot r \cdot r) = O(nr)$ time and storage over all pieces. To compute the bisectors for each pair of sites, we perform $O(r)$ updates to the corresponding persistent search tree, for a total of $\tilde{O}(nr)$ time and storage.

4 Computing Voronoi vertices

In this section we describe how to compute the vertices in the Voronoi diagram of three sites, $r, g, b \in S$, with additive weights $\omega(r), \omega(g), \omega(b)$, in $\tilde{O}(1)$ time. We also describe how to extend this computation, so that it can find the (at most two) Voronoi vertices in the diagram of $\{r, g\} \cup B$, where B is any set of sites on the boundary of a single hole of P that are determined by r, g , and some site in B .

We begin with the case of only three sites. In this case, the Voronoi diagram has at most three cells. We call a face f of P *trichromatic* if it has incident vertices in three different cells of the Voronoi diagram. *Monochromatic* and *bichromatic* faces are defined similarly. (This definition also includes faces that are holes.) We say that a vertex v of P is *red*, *green*, or *blue* if v is in the Voronoi cell of r, g , or b , respectively. By construction, the Voronoi vertices that we seek are precisely those dual to the trichromatic faces of P . Let $\beta^*(g, b)$ denote the bisector of g and b (with respect to the additive weights $\omega(g), \omega(b)$). By Lemma 2.4, $\beta^*(u, v)$ is a simple cycle in P^* . For $c \in \{g, b\}$, and for each vertex $v \in V(P)$, define $\delta^{rc}(v) := \omega(c) + d_P(c, v) - d_P(r, v)$. (These are the same as the δ^{rc} 's define in Section 3, shifted by $\omega(c)$.) Define $\Delta^r(v) := \min\{\delta^{rg}(v), \delta^{rb}(v)\}$. That is, $\Delta^r(v)$ is the maximum weight that can be assigned to r so that v is red (and v will be red also for any smaller assigned weight). For each edge uv of P , define $\Delta^r(uv) := \max\{\Delta^r(u), \Delta^r(v)\}$. That is, $\Delta^r(uv)$ is the maximum weight that can be assigned to r so that at least one endpoint of uv is red. For an edge e^* dual to a primal edge e , we put $\Delta^r(e^*) = \Delta^r(e)$.

For any $-\infty < x < \infty$, we denote by VD_x the Voronoi diagram of r, g, b , with respective additive weights $x, \omega(g), \omega(b)$. That is $VD_x = VD(\{r, g, b\}, \{x, \omega(g), \omega(b)\})$. We define

$$Q_{\geq x}^* := \{e^* \in \beta^*(g, b) \mid \Delta^r(e^*) \geq x\},$$

and we define $Q_{=x}^*$ analogously. The following lemma proves that $Q_{\geq x}^*$ is a subpath of $\beta^*(g, b)$.

Lemma 4.1. *For any $-\infty < x < \infty$, the edges of $Q_{\geq x}^*$ form a subpath of $\beta^*(g, b)$. Furthermore, if $Q_{\geq x}^*$ is non-empty and does not contain all the edges of $\beta^*(g, b)$, then the trichromatic faces in VD_x are the duals of the endpoints of $Q_{\geq x}^*$; that is, these are the faces that have exactly one incident edge in $Q_{\geq x}^*$.*

Proof. Let Q^* be a maximal subset of edges in $Q_{\geq x}^*$ that form a (contiguous) subpath of $\beta^*(g, b)$. Assume that there exists at least one edge of $\beta^*(g, b)$ that is not in Q^* ; otherwise $Q^* = \beta^*(g, b)$, and the lemma trivially holds.

Enumerate the dual vertices incident to the edges of Q^* as $f_1^*, f_2^*, \dots, f_j^*$ in their (cyclic) order along $\beta^*(g, b)$. The vertex f_1^* has an incident edge $f_1^* f_2^*$ in $Q_{\geq x}^*$, and another incident edge, call it $f_0^* f_1^*$, that is in $\beta^*(g, b)$ but not in $Q_{\geq x}^*$. In the primal, the face f_1 , dual to f_1^* , has an incident edge uv that is dual to $f_1^* f_2^*$, such that, by construction, at least one of u, v is red, and another incident edge $u'v'$, dual to $f_0^* f_1^*$, such that none of u', v' is red (note that when f_1 is a triangle, one of u', v' coincides with one of u, v). Moreover, since $f_0^* f_1^*$ is an edge of the bisector $\beta^*(g, b)$, exactly one of u', v' is blue and the other one is green. Therefore, the face f_1 is trichromatic. A similar argument shows that f_j is also trichromatic. See Figure 1.

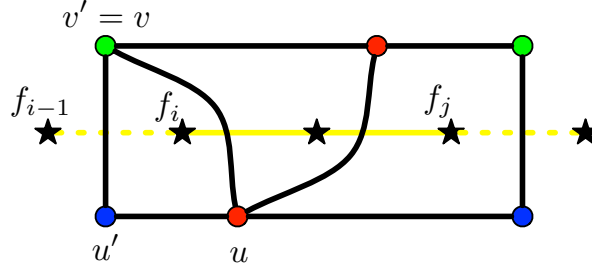


Figure 1: Illustration of the proof of Lemma 4.1. A set of consecutive edges in $\beta^*(g, b)$ is shown in yellow. Edges of Q^* are solid, and edges not in Q^* are dashed. Dual vertices are indicated by stars. Primal vertices (circles) are colored according to the Voronoi region they belong to. Since $\beta^*(g, b)$ is the green-blue bisector, the upper set of vertices is not blue, and the bottom set of vertices are not green. Primal edges are shown in black. In this example u, u' are distinct vertices, but $v' = v$. Since $f_i^* f_{i+1}^*$ is in Q^* , at least one of u and v is red. Since $f_{i-1}^* f_i^*$ is not in Q^* , neither u' nor v' is red. The face f_i is trichromatic. Similarly, f_j , the other endpoint of Q^* , is also trichromatic. Note that the argument does not rely on the faces being triangles, so it applies in the presence of holes.

This shows that every maximal subset of edges in $\Delta_{\geq x}^r$ that forms a subpath of $\beta^*(g, b)$ gives rise to two distinct trichromatic faces. Since, by Lemma 2.3, there are at most two trichromatic faces we must have that $Q^* = Q_{\geq x}^*$ and the lemma follows. \square

Recall that a linear sequence is strictly (weakly) *bitonic* if it consists of a strictly (weakly) increasing sequence followed by a strictly (weakly) decreasing sequence. A cyclic sequence is strictly (weakly) bitonic if there exists a cyclic shift that makes it strictly (weakly) bitonic. We define Δ_{β}^r to be the cyclic sequence $(\Delta^r(e^*) \mid e^* \in \beta^*(g, b))$. An immediate consequence of Lemma 4.1 is:

Corollary 4.2. *The cyclic sequence Δ_{β}^r is weakly bitonic.*

Proof. By Lemma 4.1, $Q_{\geq x}^*$ is a subpath of $\beta^*(g, b)$ and by definition we have that $Q_{\geq x'}^* \subseteq Q_{\geq x}^*$ for $x' \geq x$. \square

We will use binary search on Δ_{β}^r to find the endpoints of $Q_{\geq \omega(r)}^*$, which, by the second part of Lemma 4.1, are the trichromatic faces of the Voronoi diagram of r, g, b with respective additive weights $\omega(r), \omega(g), \omega(b)$. The search might of course fail to find these vertices when they do not exist, either because $\omega(r)$ is too small, in which case $\text{Vor}^*(r)$ completely “swallows” $\beta^*(g, b)$, or because $\omega(r)$ is too large, in which case $\text{Vor}^*(r)$ is disjoint from $\beta^*(g, b)$.

Searching in a bitonic sequence. We first briefly discuss a general strategy for conducting binary search on cyclic bitonic sequences. Given a value y , one can find the two “gaps” in a cyclic strictly bitonic sequence σ that contain y , where each such gap is a pair of two consecutive elements of σ such

that y lies between their values. (For simplicity of presentation, and with no real loss of generality, we only consider the case where y is not equal to any element of the sequence.) This is done by the following variant of binary search. We first linearize σ by cutting it at some arbitrary element, whose value is denoted as x_0 . The resulting linear sequence σ_0 is in general “tritononic”, either consisting of a contiguous increasing subsequence, followed by a contiguous decreasing subsequence, followed by a contiguous increasing subsequence, or has a similar decomposition, where the subsequences are decreasing, increasing, and decreasing, respectively. Assume, without loss of generality, that the former case arises (one can trivially check which of the two cases holds), and denote the three subsequences of σ as $\sigma_1, \sigma_2, \sigma_3$, respectively. If $y > x_0$ (resp., $y < x_0$) then the gaps that we seek belong to σ_1 and σ_2 (resp., to σ_2 and σ_3), respectively. (Note though that we do not know the places in σ that delimit these subsequences.) Assume, for specificity, that $y > x_0$.

The search consists of two phases. In the first phase the interval that the binary search maintains still contains both gaps (if they exist), and in the second phase we have already managed to separate between them, and we conduct two separate binary searches to identify each of them.

Consider a step where the search examines a specific entry $x = \sigma(i)$ of σ .

- (i) If $x < x_0$, we proceed to the left of i . (This rule holds for both phases.)
- (ii) If $x_0 \leq x < y$, we compute the discrete derivative of σ at i . If the derivative is positive (resp., negative), we proceed to the right (resp., left) of i . (This rule too holds for both phases.)
- (iii) If $x > y$ and we are in the first phase, we have managed to separate the two gaps, and we move on to the second phase with two searches, one in the current portion of σ to the left of i , and one to the right. If we are already in the second phase, the value of the discrete derivative determines in which side of i to continue the search (where in the left search we proceed in one direction, and in the right search we proceed in the opposite direction).

This procedure does not work for a weakly bitonic sequence (consider, e.g., a sequence all of whose elements, except for one, are equal). This difficulty can be overcome if, given an element i in σ such that $\sigma(i) = x$, we can efficiently find the endpoints of the maximal interval of σ that contains i and all its elements are of value equal to x (note that in general σ might contain up to two intervals of elements of value equal to x , only one of them contains i). We can then compute the discrete derivative at the output endpoint, and use it to guide the search, similar to the manner just described.

Unfortunately, given an edge $\hat{e}^* \in \beta^*(g, b)$ such that $\Delta^r(\hat{e}^*) = x$, we do not know how to find the maximal interval I of $\beta^*(g, b)$ such that $\hat{e}^* \in I$ and for every edge $e^* \in I$, $\Delta^r(e^*) = x$. Instead, we provide a procedure that returns an interval I of $\beta^*(g, b)$ that contains \hat{e}^* and also contains every other edge $e^* \in \beta^*(g, b)$ for which $\Delta^r(e^*) = x$, but it may also contain edges e^* for which $\Delta^r(e^*) > x$. Furthermore, if e_1^* and e_2^* are the first and the last edges of I , respectively, then either $\Delta^r(e_1^*) = x$ or $\Delta^r(e_2^*) = x$. If $\Delta^r(e_1^*) = x$ and $\Delta^r(e_2^*) > x$ then we know how to continue the search since we know that the Δ^r values increase for edges to the right of I . Similarly, if $\Delta^r(e_1^*) > x$ and $\Delta^r(e_2^*) = x$ we also know how to continue the search. However, if both $\Delta^r(e_1^*) = x$ and $\Delta^r(e_2^*) = x$, then Δ_β^r may be decreasing to the left of I and increasing to the right of I and then we cannot determine if the edges with Δ^r values larger than x are to the left or to the right of e^* . To resolve this ambiguity we provide a procedure that finds an edge $e_{\max}^* \in \beta^*(g, b)$ with a largest value of Δ^r . In case of uncertainty in the search, as above, e_{\max}^* must be contained in I and by checking whether e^* is to the right or to the left of e_{\max}^* we will determine on which side of e^* to continue the search.

In Section 4.1 we describe the procedure that finds an edge $e_{\max}^* \in \beta^*(g, b)$ with a largest value of Δ^r . In Section 4.2 we introduce a new definition and describe some additional preprocessing that

are required for the mechanism for finding I with the properties described above, This mechanism is described in Section 4.3.

4.1 Finding the maximum in Δ_β^r

We now describe how to find $x_{\max} := \max(\Delta_\beta^r)$, or more precisely, to find some edge $e_{\max}^* \in \beta^*(g, b)$ such that $\Delta^r(e_{\max}^*) = x_{\max}$. Let h_r be the hole to which site r is incident. We check whether r belongs to $\text{Vor}(g)$ or to $\text{Vor}(b)$ in $VD(g, b)$ by comparing the distances from g to r and from b to r . Assume that r belongs to $\text{Vor}(g)$ (the case that r belongs to $\text{Vor}(b)$ is symmetric). Then, for any arc e^* of $\beta^*(g, b)$ with $\Delta^r(e^*) = x_{\max}$ it holds that $\Delta^r(e^*) = \delta^{rg}(w)$, where w is the tail of the primal arc e . I.e, the arc e becomes red at critical value x_{\max} because its endpoint closer to g becomes red.

Let T_g^* be the cotree of the shortest path tree T_g . Define the label $\ell_g^{h_r}(f^*)$, for each dual vertex f^* to be equal to the number of edges on the f^* -to- h_r^* path in T_g^* . Note that, because the number of holes is constant, these labels can be computed in preprocessing when we compute the tree T_g^* , without changing the asymptotic preprocessing time.

Lemma 4.3. *The value of $\Delta^r(\cdot)$ is x_{\max} for at least one of the two arcs of $\beta^*(g, b)$ incident to the dual vertex $f^* \in \beta^*(g, b)$ minimizing $\ell_g^{h_r}(\cdot)$.*

Proof. Consider the dual vertex $f^* \in \beta^*(g, b)$ minimizing $\ell_g^{h_r}(\cdot)$. First assume that f^* is not a hole, so the degree of f^* is 3. Let e_1^*, e_2^* be the two arcs of $\beta^*(g, b)$ incident to f^* . Let $e_i = u_i v_i$ be the corresponding primal edges. Let e_3^* be the third arc incident of f^* . By the choice of f^* , the first arc of the f^* -to- h_r^* path in T_g^* cannot be an arc of $\beta^*(g, b)$, so this edge must be e_3^* . Since the degree of f^* is 3, it must be that $e_3 = u_1 u_2$, and that $v_1 = v_2$. See Figure 2.

Consider the fundamental cycle C of e_3 w.r.t. T_g . The cycle C encloses r , and consists entirely of edges in the cell of g in $VD(g, b)$. Hence, for every primal arc $e = wy$ such that $e^* \in \beta^*(g, b)$ (i.e., for every $w \in \beta_g$, where β_g is the cycle defined in Lemma 4.1), the r -to- w shortest path must intersect C . Let w be a vertex of β_g such that $\delta^{rg}(w) = x_{\max}$. Since r is in $\text{Vor}(g)$, by the discussion immediately preceding the lemma, w exists. Let z be an intersection vertex of the r -to- w shortest path and C . Thus, $\delta^{rg}(z) \geq x_{\max}$. Observe that, by definition of C , z is an ancestor of either u_1 or of u_2 in T_g . Assume without loss of generality that z is an ancestor of u_1 . Hence, by Lemma 4.5, $\delta^{rg}(u_1) = \Delta^r(u_1) \geq \delta^{rg}(z) = x_{\max}$. But then $\Delta^r(e_1^*) = \Delta^r(e_1) = \Delta^r(u_1) \geq x_{\max}$, so it must be that $\Delta^r(e_1^*) = x_{\max}$.

The case where f^* is a hole is similar. Instead of considering the fundamental cycle of e_3 in T_g one needs to consider the cycle formed by the root-to- u_1 path, the root-to- u_2 path, and the u_1 -to- u_2 subpath of the face f that belongs to $\text{Vor}(g)$. \square

To summarize, to find an arc of $\beta^*(g, b)$ maximizing $\Delta^r(\cdot)$ the algorithm does the following: (1) finds the site $c \in \{g, b\}$ closer to r , (2) finds the dual vertex f^* minimizing $\ell_c^{h_r}(\cdot)$ on $\beta^*(g, b)$ in $O(\log r)$ time, and (3) returns the arc of $\beta^*(g, b)$ incident to f^* with larger $\Delta^r(\cdot)$.

We next consider the same problem in the more general setting of two individual sites, say r and g , and a group $B = \{b_1 \dots, b_k\}$ of sites which are consecutive on the boundary of a hole h_b (ignoring r and/or g if they are also on h_b). We compute at preprocessing for each site u , and for each hole h , an RMQ data structure over the sites incident to h where the value of each site on h is its distance to u . This takes $O(r)$ time and does not asymptotically change the preprocessing time. We use the RMQ data structure of r and h_b to find the site, say b_i , in B to which r is the closest. We compare the distance from b_i to r and from g to r .

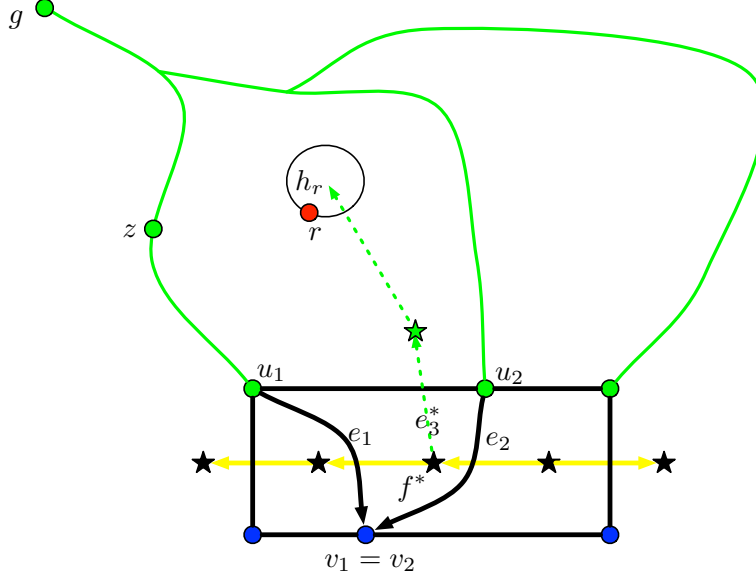


Figure 2: Illustration of the proof of Lemma 4.3. Part of $\beta^*(g, b)$ is shown in yellow. Primal vertices are represented by circles. Green circles belong to $\text{Vor}(g)$ and blue to $\text{Vor}(b)$. Dual vertices are indicated by stars. The face f^* minimizing $\ell_g^{h_r}(\cdot)$ is indicated, as well as the edges e_1, e_2 , and their endpoints. Parts of the tree T_g are shown in solid green. The only part of T_g^* shown is the f^* -to- h^* path (dashed green). As the additive weight x of the red site r decreases, the Voronoi cell of r expands. The node z is the first node on the fundamental cycle of e_3^* that enters the Voronoi cell of r . This happens at critical values x_{max} . At that time u_1 becomes red, so $\Delta^r(u_1) = x_{max}$.

Consider first the case where the distance from g to r is smaller. We traverse the segments of the bisectors $\beta^*(g, b_j)$ that form $\beta^*(g, B)$. We find the dual vertex f^* minimizing $\ell_g^{h_r}(\cdot)$ over all vertices of $\beta^*(g, B)$ by finding the minimum such vertex in each of the segments of the bisectors $\beta^*(g, b_j)$ that form $\beta^*(g, B)$. The vertex f^* depends only on the hole h_r to which r is incident and not on the particular site r itself. Therefore compute f^* once per hole and store it with the bisector $\beta^*(g, B)$ in the Voronoi cell enclosed by $\beta^*(g, B)$. The time it takes to compute f^* is dominated by the time it took to compute $\beta^*(g, B)$. As in the case of single sites we return the arc of $\beta^*(g, b)$ incident to f^* with larger $\Delta^r(\cdot)$.

Consider the case where the distance from b_i to r is smaller than the distance from g to r . Let B' be the subsequence of B (ordered by the cyclic order around h_b) consisting of all sites b' such that there is a segment of $\beta^*(g, b')$ in $\beta^*(g, B)$. Since shortest paths are non-crossing the cyclic order of these segments along $\beta^*(g, B)$ is consistent with the cyclic order of the sites of B' along h_b . Consider the case where $b_i \in B'$. Let γ^* be the segment of $\beta^*(g, b_i)$ contained in $\beta^*(g, B)$. We find the dual vertex f^* minimizing $\ell_c^{h_r}(\cdot)$ on γ^* . Let f_1^* and f_2^* be the two endpoints of γ^* . We return the arc of $\beta^*(g, B)$ incident to one of f^*, f_1^* and f_2^* with larger $\Delta^r(\cdot)$. Now consider the case where $b_i \notin B'$. In this case we find the predecessor b_p and the successor b_s of b_i in B' . Let f^* be the common dual vertex on the segment of $\beta^*(g, b_p)$ and $\beta^*(g, b_s)$ in $\beta^*(g, B)$. We return the arc of $\beta^*(g, B)$ incident to f^* with larger $\Delta^r(\cdot)$.

4.2 Arc labels

Let $c \in S$ be a site, and let T_c be the shortest path tree rooted at c . We define *labels* for the arcs of P with respect to the site c , as follows. For the sake of definition only, we modify P as follows. For each edge uv of P , we create two artificial vertices u_{uv} and v_{uv} , and two artificial arcs uv_{uv} and vu_{uv} . The arc uv_{uv} (resp., vu_{uv}) is embedded so that it immediately precedes uv (resp., the reverse of uv) in the counterclockwise cyclic order of arcs around u (resp., v). See Figure 3. (We apply this construction only once for each pair of anti-parallel arcs.) Let T'_c be the tree obtained by adding to T_c all the artificial arcs, all of which are leaf arcs in T'_c . For each arc $uv \in P$, define its *label* $\text{pre}_c(uv)$ to be the preorder index of the artificial vertex v_{uv} in T'_c , in a *CCW-first* search of T'_c .³ Similarly, define the label $\text{pre}_c(vu)$ of the reverse arc vu to be the preorder index of the vertex u_{uv} in T'_c . We define $\text{pre}_c(e^*)$ to be $\text{pre}_c(e)$. The goal of the artificial arcs and vertices is to enable us to extend the definition of the preorder labels to all arcs of P and their reverses. Note that this order is consistent with the usual preorder on just the arcs of T_c .

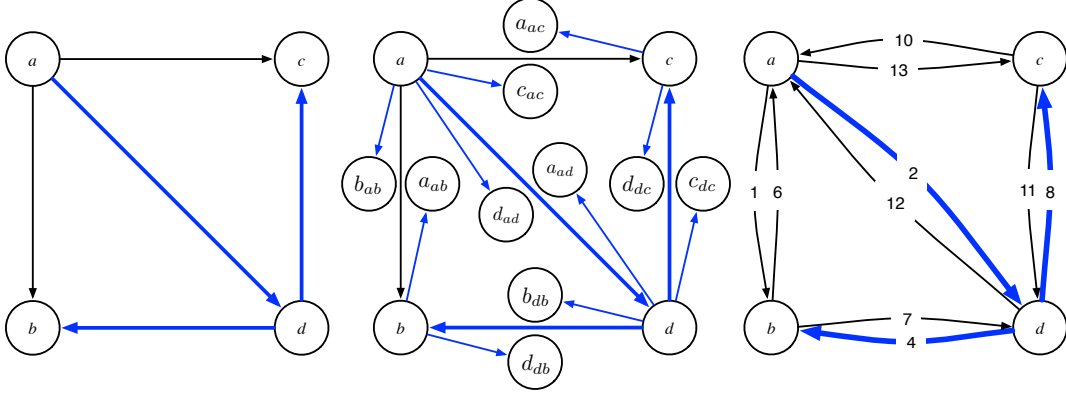


Figure 3: Illustration of the definition of labels $\text{pre}_c(\cdot)$ for all arcs. Left: A directed graph with a spanning tree T_a (in blue), rooted at vertex a . Middle: The same graph with artificial arcs and vertices. The tree T'_a is shown in blue. Right: The labels $\text{pre}_a(\cdot)$ for all arcs.

Lemma 4.4. *Let $g, b \in S$ be sites and let $c \in \{g, b\}$ be a site. Let u be a node in the cell $\text{Vor}(c)$ in $VD(\{g, b\}, \{\omega(g), \omega(b)\})$. Let p be the parent of u in T_c . Then the following hold:*

1. *The arcs in $R_u^* := \{e^* \in \beta^*(g, b) \mid \text{pre}_c(pu) < \text{pre}_c(e^*) < \text{pre}_c(up)\}$ form a subpath of $\beta^*(g, b)$.*
2. *The labels $\text{pre}_c(e^*)$ are strictly monotone along R_u^* .*

Proof. Consider the cells $\text{Vor}(g)$ and $\text{Vor}(b)$ of g and b in $VD(\{g, b\}, \{\omega(g), \omega(b)\})$. They classify the faces of P into three types: (i) those that are not incident to any vertex of $\text{Vor}(g)$, (ii) those that are not incident to any vertex of $\text{Vor}(b)$, and (iii) those that are incident to a vertex of each set. Faces of type (iii) are the faces dual to the vertices of $\beta^*(g, b)$. We denote by f_g the union of the faces of type (i) and (iii), and by f_b the union of the faces of type (ii) and (iii). It follows that for every arc $e^* \in \beta^*(g, b)$, such that $e = vw$ (so v belongs to $\text{Vor}(g)$), both v and w lie in the face f_g , where v lies on its boundary and w in its interior. See Figure 4 for an illustration. Viewed as

³A CCW-first search is a DFS that always visits the unvisited child of a node u that lies most counter-clockwise with respect to the other unvisited children, where the counter-clockwise order starts and ends at the incoming edge of u .

sets of edges in P , ∂f_g and ∂f_b are cycles, which we denote by β_g and β_b , respectively. Note that the arcs with tail on β_g and head on β_b are exactly the dual arcs of $\beta^*(g, b)$. Note also that, since the restriction of T_g to $\text{Vor}(g)$ is a connected subtree of T_g (Lemma 2.1), a branch of T_g that enters f_g (through β_g) does not leave it.

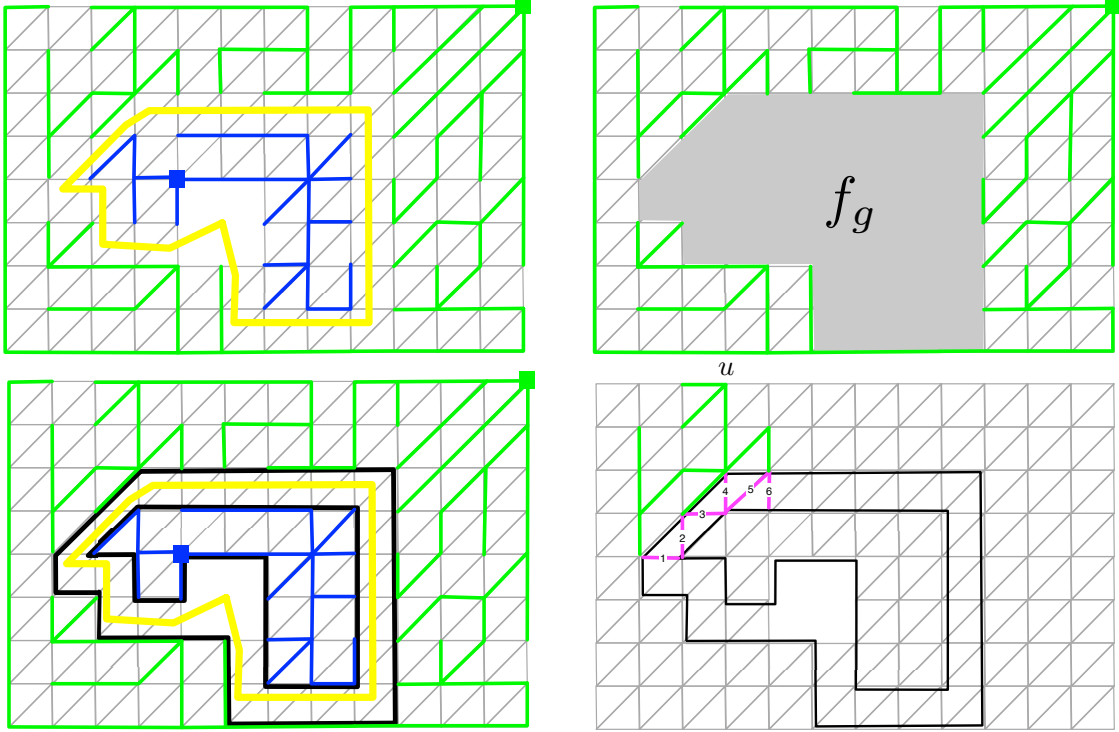


Figure 4: Illustration of the proof of Lemma 4.4. Top left: A graph P with two sites g, b . The bisector $\beta^*(g, b)$ is shown in yellow. The subtrees of T_g and T_b spanning the nodes of $\text{Vor}(g)$ and $\text{Vor}(b)$ are shown in green and blue, respectively. Top right: The Voronoi region $\text{Vor}(g)$ is shown. The face f_g is grey. Bottom left: The cycles β_g and β_b corresponding to the boundaries of f_g in $\text{Vor}(g)$ and of f_b in $\text{Vor}(b)$, respectively, are indicated in black. Bottom right: A node $u \in \text{Vor}(g)$ is indicated along with the subtree of T_g rooted at u . The primal edges of R_u^* are shown in magenta. Their relative order with respect to the labels $\text{pre}_g(\cdot)$ is indicated.

Assume, without loss of generality, that $c = g$. Consider the pair of arcs $e_1^*, e_3^* \in \beta^*(g, b)$ such that $\text{pre}_g(pu) < \text{pre}_g(e_1) < \text{pre}_g(e_3) < \text{pre}_g(up)$ (that is, $e_1^*, e_3^* \in R_u^*$), so that $\text{pre}_g(e_1)$ is smallest and $\text{pre}_g(e_3)$ is largest under the above constraints. Denote $e_i = v_i w_i$, for $i = 1, 3$, (so v_i is the endpoint in $\text{Vor}(g)$). By definition of preorder, both v_1 and v_3 are descendants of u in T_g . Consider the (undirected) cycle γ formed by the u -to- v_1 path in T_g , the u -to- v_3 path in T_g , and one of the two v_1 -to- v_3 subpaths of β_g , chosen so that γ does not enclose the blue site b . Let e_2^* be any edge in R_u^* , with a primal edge $e_2 = v_2 w_2$, where $v_2 \in \text{Vor}(g)$. Then $\text{pre}_g(pu) < \text{pre}_g(e_1) \leq \text{pre}_g(e_2) \leq \text{pre}_g(e_3) < \text{pre}_g(up)$, so v_2 is also a descendant of u in T_g , which is visited in between the visits of v_1 and v_3 . By definition of the CCW-first search preorder, at the node where the paths in T_g to v_1 and to v_2 (resp., to v_2 and to v_3) bifurcate, the arc towards v_2 lies clockwise to the arc towards v_1 (resp., counterclockwise to the arc towards v_3). Since no branch of T_g exits f_g , and no pair of paths in T_g cross each other, it follows that v_2 lies between v_1 and v_3 on $\beta_g \cap \gamma$. This establishes (1). A similar, slightly modified argument, applied to any pair of edges

e_1, e_3 satisfying $\text{pre}_g(pu) < \text{pre}_g(e_1) < \text{pre}_g(e_3) < \text{pre}_g(up)$, implies (2). \square

4.3 The mechanism

We now return to presenting our mechanism for finding I described in the beginning of Section 4. To this end, we need to exploit some structure of the shortest path trees rooted at the three sites, and the evolution of the Voronoi diagram $VD_x := VD(\{r, g, b\})$, with the weights $\omega(g), \omega(b)$ kept fixed and $\omega(r) = x$, as x decreases from $+\infty$ to $-\infty$. For $c \in \{r, g, b\}$, let $\text{Vor}_x(c)$ denote the current value of $\text{Vor}(c)$ (for the weight $\omega(r) = x$); recall that it is a subtree of T_c that spans the vertices in $\text{Vor}(c)$ in VD_x . These subtrees form a spanning forest F^x of P . We call any edge not in F^x with one endpoint in $\text{Vor}_x(c_1)$ and the other in $\text{Vor}_x(c_2)$, for a pair of distinct sites $c_1 \neq c_2$, $c_1 c_2$ -bichromatic. To handle the case where $\text{Vor}_x(r)$ is empty (e.g., at $x = +\infty$), we think of adding a super source s connected to each site c with an edge sc of weight $\omega(c)$ (in general, these edges cannot be embedded in the plane together with P), and define the edge sr to be bichromatic.

We now define some bichromatic arcs to be tense at certain critical values of x in a way similar to the definition of tense arcs in Section 3. The difference is that here an rc -bichromatic arc uv is tense at x if v gets to be closer to r at x than to **both** g and b (rather than to just one other site in Section 3). Specifically, we say that an rc -bichromatic arc uv is *tense* at x if uv is an arc of the full tree T_r , and $x + d_{T_r}(r, v) = \omega(c) + d_{T_c}(c, v) = d(s, v)$. The values of x at which some edge becomes tense are the critical values at which the rb - or the rg -bisector changes. Recall that we assume that at each critical value x only one arc is tense. At the critical value x , the red tree $\text{Vor}_x(r)$ takes over the node v of the tense arc uv . Let $\text{Vor}_x^-(c)$ be the primal Voronoi cell of c just before the critical value x . For any descendant w of v in $\text{Vor}_x^-(c)$, we have

$$d(s, w) = \omega(c) + d_{T_c}(c, v) + d_{T_c}(v, w) = x + d_{T_r}(r, v) + d_{T_c}(v, w),$$

so the red tree also takes over the entire subtree of v in $\text{Vor}_x^-(c)$ and we the following lemma follows.

Lemma 4.5. *At any critical value x of $\omega(r)$ the vertices that change color at x are precisely the descendants of v in $\text{Vor}_x^-(c)$ where uv is the unique bichromatic tense arc at x .*

Consider the sequence Δ_β^r , and let $x' > x$ be two consecutive values in it. Recall that both $\Delta_{\geq x'}^r$ and $\Delta_{\geq x}^r$ are contiguous (cyclic) subsequences of Δ_β^r (Lemma 4.1). Assume $\Delta_{\geq x'}^r = \Delta_\beta^r[j \dots k]$, and $\Delta_{\geq x}^r = \Delta_\beta^r[i \dots l]$ (where the indices are taken modulo the length of Δ_β^r). Since, by definition, $\Delta_{\geq x'}^r \subset \Delta_{\geq x}^r$, the set $\Delta_{=x}^r := \Delta_{\geq x}^r \setminus \Delta_{\geq x'}^r$ of elements of Δ_β^r with value exactly x , forms two intervals $[i \dots j), (k \dots l]$ of Δ^r , at least one of which is non-empty. Let uv be the unique bichromatic tense edge at critical value x . Let $c \in \{g, b\}$ be such that uv is rc -bichromatic. By the preceding arguments, including Lemma 4.5, the nodes w with $\Delta^r(w) = x$ are descendants of v in the subtree of T_c spanning $\text{Vor}_{x'}(c)$. Furthermore, the subtree of T_c spanning $\text{Vor}_\infty(c)$ (this is the cell of c in $VD(\{g, b\})$ when r is at distance ∞ from all vertices) contains the subtree of T_c spanning $\text{Vor}_{x'}(c)$ and additional nodes w for which $\Delta^r(w) = \delta^{rc}(w) \geq x$ that switched to the red tree at earlier critical values, higher than x (recall that x is decreasing).

Recall the definition of the path R_v^* in the statement of Lemma 4.4 (with respect to the bisector $\beta^*(g, b)$). It follows, by the discussion above, that any edge e^* of R_v^* has $\Delta^r(e^*) \geq x$, and all edges e^* with $\Delta^r(e^*) = x$ belong to R_v^* . Let e_1^* , and e_2^* be the first and last edges of R_v^* , respectively. By Lemma 4.4, e_1^* and e_2^* are also the edges with minimum and maximum values of $\text{pre}_c(\cdot)$ in R_v^* , respectively. It follows that at least one of $\Delta^r(e_1^*), \Delta^r(e_2^*)$ is x (and the other is $\geq x$), and that, moreover, either e_1^* or e_2^* is an extreme edge in the maximal interval of edges $Q_{\geq x}^*$ of $\beta^*(g, b)$.

Exploiting these properties, we design the following procedure $\text{GETINTERVAL}(e^*)$. The input is an edge $\hat{e}^* \in \beta^*(g, b)$ with $\Delta^r(\hat{e}^*) = x$. The output is an interval I of extreme edges e^* of $Q_{\geq x}^*$ such that 1) either the first or the last edge of I is of value x , 2) I contains all edges of value x in $\beta^*(g, b)$ (and \hat{e}^* in particular).

$\text{GETINTERVAL}(e^*)$

1. Let $e = vw$ be the primal edge of e^* . Find the endpoint of e whose $\Delta^r(\cdot)$ value is x (Lemma 4.5 implies that there is only one such endpoint). Suppose, without loss of generality, that $\Delta^r(v) = x$.
2. Find the Voronoi cell to which v belongs in $VD(\{g, b\}, \{\omega(g), \omega(b)\})$. Let c be the corresponding site. So $\Delta^r(v) = \delta^{rc}(v) = x$.
3. Find the ancestor u of v in T_c that is nearest to the root, such that $\delta^{rc}(u) = x$. Note that the node u is an endpoint of the unique tense edge at critical value x . Finding u can be done by binary search on the root-to- v path in T_c since, by Lemma 4.5, all the ancestors of u on this path have strictly smaller Δ^{rc} -values.
4. Let p be the parent of u in T_c . Return the interval that starts at the successor of $\text{pre}_c(pu)$ and ends at the predecessor of $\text{pre}_c(up)$ in $\beta^*(g, b)$ (with respect to the $\text{pre}_c(\cdot)$ numbering of the edges of $\beta^*(g, b)$). Since, by Lemma 4.4, the cyclic order on $\beta^*(g, b)$ is consistent with $\text{pre}_c(\cdot)$, we can find these successor and predecessor by using $\text{pre}_c(\cdot)$ as an additional key in the search tree representing $\beta^*(g, b)$.

We efficiently implement $\text{GETINTERVAL}(e^*)$ as follows. We retrieve and compare the distances from r , g , and b to v and w . This gives $\Delta^r(v)$, $\Delta^r(w)$, and thereby $\Delta^r(e)$. It also reveals the Voronoi cell in $VD(\{g, b\}, \{\omega(g), \omega(b)\})$ containing v . All this is done easily in $O(1)$ time. To carry out the binary search to find the ancestor u of v in T_c that is nearest to the root, such that $\delta^{rc}(u) = x$ we use a level ancestor data structure on T_c which we prepared at the preprocessing. Each query to this data structure takes $O(1)$ time.

Using the weak bitonicity of Δ_β^r (Corollary 4.2), the procedure GETINTERVAL , and the procedure for finding e_{\max}^* described in Section 4.1, we can find the trichromatic faces that are dual to the Voronoi vertices of $VD^*(\{r, g, b\})$, under the respective weights $\omega(r), \omega(g), \omega(b)$ by the variation of binary search described before.

Several additional enhancements of the preprocessing stage are needed to support an efficient implementation of this procedure. First, we need to store T_c for each individual site c . Second, for each bisector $\beta^*(c_1, c_2)$ we need to store in its persistent search tree representation two secondary keys $\text{pre}_{c_1}(\cdot)$ and $\text{pre}_{c_2}(\cdot)$. As we discussed above, these keys are consistent with the cyclic order of $\beta^*(c_1, c_2)$. Clearly, all these enhancements do not increase the preprocessing time asymptotically.

4.4 Dealing with a group of sites on a single hole

We now consider a more general scenario, where there is a single red site r , a single green site g , and a set $B = \{b_1, \dots, b_k\}$ of k blue sites on the boundary of a single hole h (in case r and/or b also lie on ∂h , we require that r and/or g , and B be separated in the cyclic order along ∂h). The goal is to find the trichromatic faces of $VD(\{r, g\} \cup B)$, under the currently assigned weights; by a *trichromatic face* we now mean a face with at least one vertex in $\text{Vor}(r)$ (a red vertex), at least one vertex in $\text{Vor}(g)$ (a green vertex), and at least one vertex in $\text{Vor}(b_i)$, for some i (a blue vertex). Handling this situation is fairly similar to the simpler case of three individual sites, but requires

several modifications that are needed in the presence of multiple blue sites, which we discuss next. Note that, by creating a “super-blue” site inside h , and by connecting it to all the blue sites, Lemma 2.3 still holds in this setting, so, as in the simpler case, there are at most two trichromatic faces.

Next, Lemma 4.1 still holds, but its proof needs a slight modification, because Lemma 2.4 might not apply, and we need to use instead Lemma 5.2, which only guarantees that $\beta^*(g, B)$ is a non-self-crossing cycle, which might pass multiple times through h^* . The proof of Lemma 4.1 considers a maximal contiguous interval of edges of $\beta^*(g, b)$ whose primal versions have at least one red endpoint, argues that the extreme vertices in such a path must be trichromatic, and concludes the uniqueness of the path by Lemma 2.3. Here, if h^* is trichromatic, we could in principle have two edge-disjoint maximal subpaths I_1, I_2 of $\beta^*(g, B)$ that share h^* as an endpoint. Since h^* is trichromatic, there is at most one other trichromatic vertex g^* along $\beta^*(g, B)$. It is impossible that both I_1 and I_2 share g^* too, for then we could have merged them into a larger subpath with the same property, contradicting their minimality. Hence, among the four endpoints of I_1 and I_2 , at least three are h^* . This, combined with the fact that $\beta^*(g, B)$ is non-self-crossing, imply the following property: We can choose an endpoint of I_1 equal to h^* and an endpoint of I_2 equal to h^* , consider the edges e_1^*, e_2^* incident to the first endpoint, with e_1^* outside I_1 and e_2^* inside, and the edges e_3^*, e_4^* incident to the second endpoint, with e_3^* outside I_2 and e_4^* inside, so that the cyclic order of these edges around h^* is $(e_1^*, e_2^*, e_3^*, e_4^*)$. As in the proof of Lemma 4.1, for each of e_2^*, e_4^* , its primal edge has at least one red endpoint, and for each of e_1^*, e_3^* , its primal edge has one green endpoint and one blue endpoint. The two red endpoints can be connected by a path in $\text{Vor}(r)$, consisting exclusively of red vertices, and the two green endpoints can be connected by a path in $\text{Vor}(g)$, consisting exclusively of green vertices. However, by the cyclic order of these edges along ∂h , the red and the green paths must cross one another, which is impossible because $\text{Vor}(r)$ and $\text{Vor}(g)$ are disjoint. This contradiction shows that Lemma 4.1 continues to hold in this case too. See Figure 5.

We also need to slightly revise the statement of Lemma 4.4. For $c = g$ the statement and proof remain unchanged. For $c = b_i$, item (1) should be: The edges in

$$R_u^* := \{e^* \text{ in the segment of } \beta^*(g, b_i) \text{ along } \beta^*(g, B) \mid \text{pre}_c(pu) < \text{pre}_c(e^*) < \text{pre}_c(up)\}$$

is a contiguous subpath of $\beta^*(g, B)$. The proof remains the same but should be applied to each cell $\text{Vor}(b_i)$ of the diagram $VD(\{g\} \cup B)$. No changes are required in Lemma 4.5. In the procedure GETEXTREME, step 2 needs no change, just note that now the Voronoi cell is for some individual site b_i . Therefore, in step 4, find the successor and predecessor in the segment of $\beta^*(g, b_i)$ on $\beta^*(g, B)$.

5 Additional structure of Voronoi diagrams

In this section we study the structural properties that will be used in the fast construction of a Voronoi diagram from the precomputed bisectors. We first repeat the lemmas from Sections 2 and 3, this time including the proofs.

Lemma 2.1. *For each $u \in S$, the vertices in $\text{Vor}(u)$ form a connected subtree (rooted at u) of T_u .*

Proof. This is an immediate consequence of the property that shortest paths cannot cross one another (under our non-degeneracy assumption); in fact, they cannot even meet one another except at a common prefix. \square

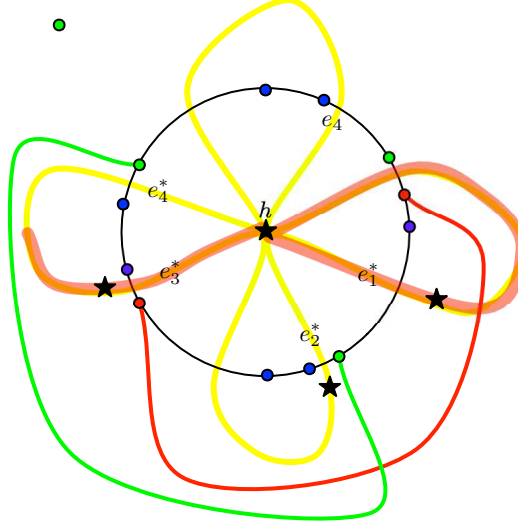


Figure 5: Modification to the proof of Lemma 4.1. Two non-consecutive subpaths of $\beta^*(g, B)$ formed by edges of $\Delta_{\geq x}^r$ are highlighted in orange. These subpaths meet at the dual vertex h . The edges $e_1^*, e_2^*, e_3^*, e_4^*$ are indicated (their endpoints (dual vertices) are shown as stars), as well as possible colors for their primal endpoints. A contradiction arises since the red path and the green path intersect.

Lemma 2.2. *The graph $VD^*(S)$ consists of at most b faces, so that each of its faces corresponds to a site $u \in S$ and is the union of all faces of P^* that are dual to the vertices of $\text{Vor}(u)$.*

Proof. For each $u \in S$, the union of all faces of P^* that are dual to the vertices of $\text{Vor}(u)$ is connected, since $\text{Vor}(u)$ is a tree. Each face of P^* belongs to exactly one face of $VD^*(S)$, because the trees $\text{Vor}(u)$ are pairwise disjoint, and form a partition of V . Hence the faces of $VD^*(S)$ stand in 1-1 correspondence with the trees $\text{Vor}(u)$, for $u \in S$, and the claim follows. \square

Lemma 2.3. *For any three distinct sites u, v, w in S there are at most two faces f of P such that each of the cells $\text{Vor}(u), \text{Vor}(v), \text{Vor}(w)$ contains a vertex of f .*

Proof. Assume to the contrary that there are three such faces f_1, f_2, f_3 . Let p_i, q_i, r_i , for $i = 1, 2, 3$, denote the vertices of f_i satisfying $p_i \in \text{Vor}(u)$, $q_i \in \text{Vor}(v)$, and $r_i \in \text{Vor}(w)$. Let f_i^* be an arbitrary point inside f_i , for $i = 1, 2, 3$. We construct the following embedding of $K_{3,3}$ in the plane. One set of vertices is $\{f_1^*, f_2^*, f_3^*\}$ and the other is $\{u, v, w\}$. To connect f_i^* with u , say, we connect u to p_i via the shortest path $\pi(u, p_i)$, concatenated with the segment $p_i f_i^*$. The connections with v, w are drawn analogously. It is easily verified that the edges in this drawing do not cross one another (because shortest paths do not cross one another), so we get an impossible planar embedding of $K_{3,3}$, a contradiction that implies the claim. See Figure 6. \square

Lemma 2.4. *$\beta^*(u, v)$ is a simple cycle of arcs of P^* . If u and v are incident to the same hole h and $\beta^*(u, v)$ is nonempty then $\beta^*(u, v)$ is incident to h^* .*

Proof. $\text{Vor}(u)$ and $\text{Vor}(v)$ form a partition of the vertices of P into two connected sets. Therefore, the set B of arcs with tail in $\text{Vor}(u)$ and head in $\text{Vor}(v)$ is a simple cut in P . By the duality of simple cuts and simple cycles, $B^* = \beta^*(u, v)$ is a simple cycle.

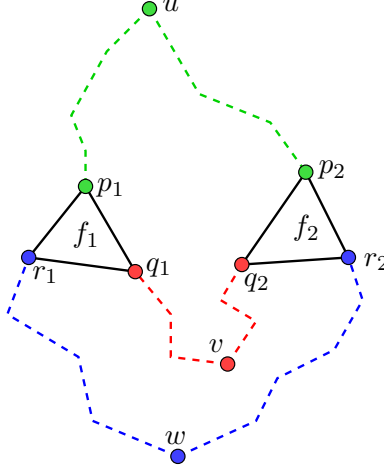


Figure 6: Illustration of the proof of Lemma 2.3.

If u and v are incident to the same hole and $\beta^*(u, v)$ is nonempty, the simple cut defined by the partition $(\text{Vor}(u), \text{Vor}(v))$ must contain an arc e on the boundary of h . Therefore, e^* is an arc of $\beta^*(u, v)$ that is incident to h^* . \square

Lemma 3.1. *Consider some critical value δ . The dual edges that newly join $\beta^*(u, v)$ at δ form a contiguous portion of the new bisector, and the dual edges that leave $\beta^*(u, v)$ at δ form a contiguous portion of the old bisector.*

Proof. Let $\text{Vor}^-(u)$ be the primal Voronoi cell of u right before δ and let $\text{Vor}^+(v)$ be the primal Voronoi cell of v right after δ . Let yz be the tense edge that triggers the switch, so z is the root of the subtree that moves from $\text{Vor}^-(u)$ to $\text{Vor}^+(v)$ at δ . Let $\beta^{*-}(u, v)$ and $\beta^{*+}(u, v)$ denote, respectively, the uv -bisector immediately before and after δ . By Lemma 4.4 the preorder numbers (in T_u say) of the edges along $\beta^{*-}(u, v)$ are monotonically increasing and therefore the arcs of $\beta^{*-}(u, v)$ whose tails are in the subtree of z must form a continuous portion of $\beta^{*-}(u, v)$. An analogous argument applies to the arcs of $\beta^{*+}(u, v)$ whose heads are in the subtree of z . \square

We next continue with additional properties and definitions.

Lemma 5.1. *Let f^* and g^* be two Voronoi vertices of $VD^*(S)$ which are consecutive on the common boundary between the cells $\text{Vor}^*(u)$ and $\text{Vor}^*(v)$. Then the path between f^* and g^* along this boundary in $VD^*(S)$ is a (connected) segment of $\beta^*(u, v)$.*

Proof. If $xy \in B$ where $x \in \text{Vor}(u)$ and $y \in \text{Vor}(v)$ then x is closer to u than to v and y is closer to v than to u . It follows that $x \in \text{Vor}(u)$ and $y \in \text{Vor}(v)$ also when S contains only the two sites u and v . \square

We generalize the notion of bisectors to sets of sites. Let h_g, h_b be (not necessarily distinct) holes. Let $G \subset S$ be a set of “green” sites incident to h_g and let $B \subset S$ be a set of “blue” sites incident to h_b ; when $h_g = h_b$ we require that G and B be separated along ∂h_g .

We define $\beta^*(G, B)$ to be the set of edges of P^* whose corresponding primal arcs have their tail in $\text{Vor}(g_i)$ and head in $\text{Vor}(b_j)$ for some $g_i \in G$, and $b_j \in B$.

Lemma 5.2. *If $h_g \neq h_b$ or if $h_g = h_b$, but the sets G, B are separated along the boundary of h_g , then $\beta^*(G, B)$ is a non-self-crossing cycle of arcs of P^* . If $|G| > 1$ then h_g^* may have degree higher than 2 in $\beta^*(G, B)$ and if $|B| > 1$ then h_b^* may have degree higher than 2 in $\beta^*(G, B)$. All other dual vertices have degree 0 or 2 in $\beta^*(B, G)$. Furthermore, if $h_g = h_b$ and if $\beta^*(G, B)$ is nonempty, then $\beta^*(G, B)$ contains h_g^* (possibly multiple times).*

Proof. Embed a “super-green” vertex g inside h_g , and a “super-blue” vertex b inside h_b . This can be done without violating planarity also when $h_g = h_b$, since in this case G, B are separated along ∂h_g . Connect h_g (resp., h_b) to the green (resp., blue) sites with arcs gg_i (resp., bb_i) of weight $\omega(g_i)$ (resp., $\omega(b_i)$). By Lemma 2.4, the bisector $\beta^*(g, b)$ is a simple cycle in this auxiliary graph. However, this cycle can go through the artificial faces created in the holes h_g, h_b . Deleting the artificial arcs in the primal is equivalent to contracting them in the dual, which contracts all the artificial faces into the dual faces h_g^* and h_b^* . This contraction gives rise to non-simplicities at h_g^* and h_b^* . See Figure 7.

The proof of the final property is identical to the one in Lemma 2.4. Namely, If $h_g = h_b$ and $\beta^*(B, G)$ is nonempty, then the simple cut defined by the partition $(\text{Vor}(g), \text{Vor}(b))$ must contain an arc e on the boundary of h_g . Therefore, e^* is an arc of $\beta^*(u, v)$ that is incident to h_g^* . \square

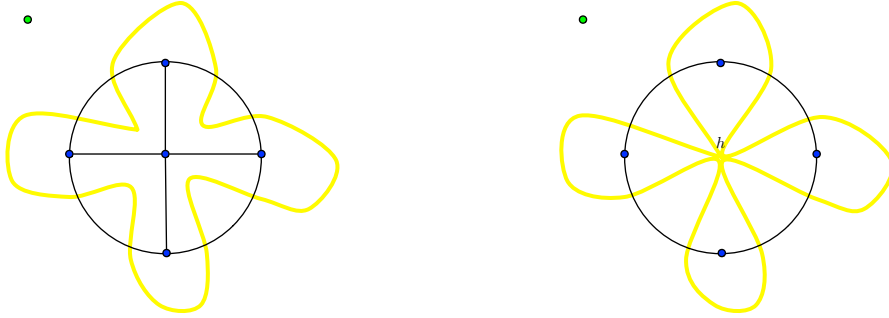


Figure 7: The structure of the bisector $\beta^*(G, B)$ when $|G| = 1$. Left: a set B of four blue sites on the hole h (black cycle). An artificial blue site b is embedded in h and connected to all blue sites. The bisector $\beta^*(g, b)$ is a simple cycle. Right: When removing the artificial arcs the bisector $\beta^*(g, B)$ visits the face h multiple times.

6 Computing the Voronoi diagram

Representing the diagram. We represent $VD^*(S)$ by a reduced graph in which we contract all the vertices of degree 2. That is, we replace each path $p^* = p_1^*, p_2^*, \dots, p_l^* = q^*$ in $VD^*(S)$, where p^* and q^* are Voronoi vertices, and $p_2^*, p_3^*, \dots, p_{l-1}^*$ are vertices of degree 2, by the single edge (p^*, q^*) . We represent this reduced graph using the standard DCEL data structure for representing planar maps (see [12] for details). Note that, in degenerate situations, $VD^*(S)$ might contain an isolated loop that consists of a single bisector and has no Voronoi vertices on it. In this case the diagram is disconnected, and structures representing the cells (rather than the edges) connect between different components of the boundary of non-simply-connected cells. By Lemma 5.1, the path $(p^* = p_1^*, p_2^*, \dots, p_l^* = q^*)$ is a contiguous portion of some bisector $\beta^*(u_1, u_2)$. We store with the contracted edge (p^*, q^*) pointers that (a) identify the sites u_1, u_2 whose dual cells are adjacent

to this edge, and (b) point to the first and the last edges of this portion, namely to the edges (p^*, p_2^*) and (p_{l-1}^*, q^*) .

We describe an algorithm that, given access to the pre-computed representations of the bisectors, computes $VD^*(S)$ in $\tilde{O}(b)$ time.

6.1 Single hole

The divide-and-conquer mechanism. Assume that there is only one distinguished face (hole) h_1 , and let h_1^* denote its dual vertex. We describe a divide-and-conquer algorithm for constructing $VD^*(S)$ in $\tilde{O}(b)$ time. Note that in this case the diagram does not contain an isolated loop; that is, its edges form a connected graph (see below for a more precise statement).

We partition the set S of sites into two contiguous subsets of (roughly) the same size $k \approx b/2$. Each subset is consecutive in the cyclic order around ∂h_1 . The sites in one subset, Y , denoted as y_1, \dots, y_k in their clockwise order around h_1 , are referred to as the *yellow* sites, and those of the other subset O , denoted as o_1, \dots, o_k in their clockwise order around h_1 , are the *orange* sites. We recursively compute the Voronoi diagram of Y , denoted as $VD^*(Y)$, and the Voronoi diagram of O , denoted as $VD^*(O)$. We now describe how to merge these two diagrams and obtain $VD^*(S)$, in $\tilde{O}(b)$ time.

Constructing $VD^*(Y, O)$. To merge $VD^*(Y)$ and $VD^*(O)$, we have to identify all segments of the bisectors $\beta^*(y, o)$, for $y \in Y$ and $o \in O$, that belong to $VD^*(S)$. Recall that we denote the subgraph of $VD^*(S)$ induced by the edges of these segments by $VD^*(Y, O)$. The structure of $VD^*(Y, O)$ is specified by Lemma 5.2 (this is the case where $h_g = h_b$ in the statement of the Lemma 5.2).

Our algorithm consists of two main stages. In the first stage we identify the edges of $VD^*(Y, O)$ that are incident to h_1^* ; that is, edges of $VD^*(Y, O)$ that are dual to edges on ∂h_1 . We denote this subset of $VD^*(Y, O)$ by $VD_I^*(Y, O)$. In the second stage we compute the paths that comprise $VD^*(Y, O)$, each of which connects a pair of edges of $VD_I^*(Y, O)$, which were obtained in Stage 1. Each such path is a concatenation of contiguous portions of “bichromatic” bisectors of the form $\beta^*(y_i, o_j)$.

Stage 1. We partition ∂h_1 into segments by cutting off its edges whose duals are in $VD^*(Y) \cup VD^*(O)$. The vertices in each such segment belong to a single Voronoi cell $\text{Vor}(y_i)$ of $VD(Y)$, and to a single Voronoi cell $\text{Vor}(o_j)$ of $VD(O)$, and we denote this segment by A_{ij} .

For each segment A_{ij} , we check whether one of the edges, e^* , of $\beta^*(y_i, o_j)$ that is incident to h_1^* is dual to an edge in A_{ij} . Since $\beta^*(y_i, o_j)$ is a simple cycle, it can pass through h_1^* at most once, so it has at most two edges that cross ∂h_1 . These edges can easily be retrieved from the precomputed data involving $\beta^*(y_i, o_j)$, as constructed during the sweeping procedure at preprocessing. If we have found such an edge e^* , it belongs to $VD_I^*(Y, O)$, and we add it to that set.

Another way in which an edge e^* can belong to $VD_I^*(Y, O)$ is when it is a delimiter between two consecutive segments A_{ij} and $A_{i'j'}$ (where we can have $i = i'$ or $j = j'$, but not both). Let $e = (x, x')$ be the primal edge of ∂h dual to e^* , with $x \in A_{ij}$ and $x' \in A_{i'j'}$. In this case, the site in $Y \cup O$ nearest to x must be either y_i or o_j , and the site nearest to x' must be either $y_{i'}$ or $o_{j'}$. Then e^* is an edge of $VD_I^*(Y, O)$ if and only if the site nearest to x and the site nearest to x' are of different colors (a test that we can perform in constant time).

Since the preceding arguments exhaust all possibilities, we obtain the following lemma that asserts the correctness of this stage.

Lemma 6.1. *The procedure described above identifies all edges of $VD_I^*(Y, O)$.*

Stage 2. The set $VD_I^*(Y, O)$ produced in Stage 1 consists of the edges incident to h_1^* on each cycle $h \in VD^*(Y, O)$ (by Lemma 5.2, all these cycles pass through h_1^*). Moreover, as already argued, each of these cycles contains exactly two such edges. Saying it slightly differently, ignoring h_1^* , each of these cycles is a simple path in P^* that starts and ends at a pair of respective edges of $VD_I^*(Y, O)$, and otherwise does not meet ∂h_1 . Stage 2 constructs these paths iteratively, picking an edge e_1^* of $VD_I^*(Y, O)$, and tracing the cycle γ that starts at e_1^* until it reaches ∂h_1 again, at another “matched” edge $e_2^* \in VD_I^*(Y, O)$; the stage repeats this tracing step until all the edges of $VD_I^*(Y, O)$ are exhausted.

We compute the cycle $\gamma \in VD^*(Y, O)$ containing an initial edge $e_1^* \in VD_I^*(Y, O)$ as follows. Let $\beta^*(y_i, o_j)$ be the bisector that contains e_1^* in $VD^*(Y, O)$. It follows that e_1^* is either contained in, or lies on the boundary of $\text{Vor}^*(y_i)$ in $VD^*(Y)$, and the same holds for $\text{Vor}^*(o_j)$ in $VD^*(O)$. We use the algorithm of Section 4 to find the intersection I_1 between $\beta^*(y_i, o_j)$ and $\text{Vor}^*(y_i)$ in $VD^*(Y)$, and the intersection I_2 between $\beta^*(y_i, o_j)$ and $\text{Vor}^*(o_j)$ in $VD^*(O)$; each of these intersections is a segment of the bisector between two Voronoi vertices.

Each endpoint of I_1 is a Voronoi vertex adjacent to $\text{Vor}(y_i)$, $\text{Vor}(o_j)$, and to some third cell $\text{Vor}(y_{i'})$ in $VD^*(Y \cup \{o_j\})$, and each endpoint of I_2 is a Voronoi vertex adjacent to $\text{Vor}(y_i)$, $\text{Vor}(o_j)$, and to some third cell $\text{Vor}(o_{j'})$ in $VD^*(O \cup \{y_i\})$.

We then compute $I = I_1 \cap I_2 \subseteq \beta^*(y_i, o_j)$. This portion of $\beta^*(y_i, o_j)$ is a bichromatic Voronoi edge (of the final diagram) in $VD^*(Y, O)$. If I is the entire $\beta^*(y_i, o_j)$ then $\gamma = I$ and we are done. Otherwise, each endpoint of I is a Voronoi vertex of $VD^*(Y, O)$ adjacent, in the merged diagram $VD^*(S)$, to the cells $\text{Vor}(y_i)$, $\text{Vor}(o_j)$, and to a third cell which is either $\text{Vor}(y_{i'})$ for some yellow site $y_{i'} \neq y_i$, or $\text{Vor}(o_{j'})$ for some orange site $o_{j'} \neq o_j$. Let g^* be an endpoint of I (different from h_1^* if h_1^* is an endpoint of I); assume for concreteness that g^* is adjacent in $VD^*(S)$ to $\text{Vor}(y_i)$, $\text{Vor}(o_j)$, and to some other $\text{Vor}(y_{i'})$. The next segment of γ that is adjacent to g^* is a segment I' of $\beta^*(y_{i'}, o_j)$ (the third edge adjacent to g^* in $VD^*(S)$ is a truncated portion of $\beta^*(y_i, y_{i'})$ from $VD^*(Y)$, and is not part of $VD^*(Y, O)$). We construct I' by finding the intersection I'_1 of $\beta^*(y_{i'}, o_j)$ and $\text{Vor}^*(y_{i'})$ in $VD^*(Y)$, and the intersection I'_2 of $\beta^*(y_{i'}, o_j)$ and $\text{Vor}^*(o_j)$ in $VD^*(O)$. The segment I' is the intersection of I'_1 and I'_2 . One endpoint of I' is g^* , and we continue tracing γ from the other endpoint of I' in this manner, identifying the Voronoi edges on γ one by one, until we reach the other endpoint of the initial interval I at which the tracing has started.

By construction, the initial segment I contains h_1^* , either as an endpoint or as an interior point. Therefore, the second edge e_2^* (in addition to e_1^* where we started tracing γ) of $VD_I^*(Y, O)$ contained in γ is in I if h_1^* is not an extreme vertex of I , or else it is the last edge of the last segment that we have traced. We then discard e_1^* and e_2^* from $VD_I^*(Y, O)$, and keep performing this tracing procedure from a fresh, unvisited edge, until all edges of $VD_I^*(Y, O)$ are exhausted, obtaining in this way all the cycles of $VD^*(Y, O)$.

Wrap-up. Once we have computed the cycles of $VD^*(Y, O)$, we can assemble the entire Voronoi diagram $VD^*(S)$ by pasting parts of $VD^*(Y)$ and $VD^*(O)$ to $VD^*(Y, O)$. Each vertex of $VD^*(Y, O)$ is adjacent to an edge whose segment is a subsegment of an edge of $VD^*(Y)$ or of $VD^*(O)$. We retrieve each of these monochromatic edges, split it at the respective bichromatic vertex h_1^* , and glue the appropriate portion to $VD^*(Y, O)$ at h_1^* , discarding the complementary portion of the edge. Upon completion, we get rid of the portions of $VD^*(Y)$ and of $VD^*(O)$ that are no longer part of the full diagram, by a simple graph search from the discarded edge portions. The surviving (unsplit) edges of $VD^*(Y)$ and $VD^*(O)$ are edges of $VD^*(S)$ too.

Running time. Each call to the procedure of Section 4 takes $\tilde{O}(1)$ time, and the number of calls is proportional to the overall complexity of $VD(Y)$, $VD(O)$, and $VD(S)$, which is $O(b)$. It easily follows that the overall cost of the divide-and-conquer mechanism is $\tilde{O}(b)$.

6.2 Two holes

Assume now that P has only two holes (distinguished faces) h_1 and h_2 , on which the sites are located. We call the sites on ∂h_1 (resp., on ∂h_2) the *green sites* (resp., *red sites*), and denote those subsets of S by G and R , respectively. We extend the algorithm of Section 6.1 to compute $VD^*(S)$ in this “bichromatic” case. Let h_1^* and h_2^* be the dual vertices of h_1 and h_2 , respectively.

Our algorithm first computes the Voronoi diagram $VD^*(G)$ of the green sites, and the Voronoi diagram $VD^*(R)$ of the red sites, using the algorithm of Section 6.1. (The presence of the other hole does not affect the construction of either diagram in any significant way.) We then merge $VD^*(G)$ and $VD^*(R)$ into $VD^*(S)$, by finding the set $VD^*(G, R)$ of the bichromatic edges of $VD^*(S)$, each of which corresponds to a segment of a bisector of the form $\beta^*(g, r)$ for some $g \in G$ and $r \in R$. The structure of $VD^*(G, R)$ is characterized in Lemma 5.2.

Our algorithm has two stages, similar to the two stages of the algorithm in the previous subsection. In the first stage we compute the edges of $VD^*(G, R)$ that are dual to edges of h_1 or of h_2 . We denote this subset of $VD^*(G, R)$ by $VD_I^*(G, R)$. This is done in exactly the same way as the computation of $VD_I^*(Y, O)$ in the previous section.

Note however that here, in contrast with the previous algorithm, $VD_I^*(G, R)$ is empty in case $VD^*(G, R)$ is a simple cycle which does not contain h_1^* and h_2^* . When $VD_I^*(G, R) = \emptyset$, we identify an edge e^* on the “free-floating” cycle $VD^*(G, R)$ using a special procedure that we describe below.

In the second stage we construct the whole $VD^*(G, R)$. If $VD_I^*(G, R) \neq \emptyset$ (i.e., if $\beta^*(G, B)$ is incident to h_1^* or to h_2^*), then each cycle of $VD^*(G, R)$ contains edges of $VD_I^*(G, R)$ and decomposes into paths connecting these edges. Each of these paths connects a pair of edges of $VD_I^*(G, R)$, where the two edges in such a pair either are both dual to edges of h_1 , or are both dual to edges of h_2 , or one is dual to an edge of h_1 and one dual to an edge of h_2 . We compute these paths and cycles exactly as we computed, in Section 6.1, the paths of $VD^*(Y, O)$ connecting pairs of edges of $VD_I^*(Y, O)$.

If $VD^*(G, R)$ is a simple cycle that does not contain h_1^* and h_2^* , we compute it using a similar tracing procedure, but this time starting from the edge $e^* \in VD^*(G, R)$ that we found by the special procedure.

To complete the description of the computation of $VD^*(G, R)$, it remains to describe how we identify a starting edge $e^* \in VD^*(G, R)$ when $VD^*(G, R)$ is a simple cycle that does not contain h_1^* and h_2^* .

Finding an edge $e^* \in VD^*(G, R)$ when $VD^*(G, R)$ is a simple cycle. We split this procedure into two cases.

Case 1: There is a path between h_1^* and h_2^* in $VD^*(G)$. (The case where there is a path between h_1^* and h_2^* in $VD^*(R)$ is treated symmetrically.) We test whether this is the case by checking, for each edge of $VD^*(G)$, whether it contains h_1^* or h_2^* (as already noted earlier, this information is available from the preprocessing stage). Let $\pi^*(h_1^*, h_2^*)$ be such a path in $VD^*(G)$ between h_1^* and h_2^* . The first edge of $\pi^*(h_1^*, h_2^*)$ is dual to an edge (x_1, y_1) of h_1 , and the last edge of $\pi^*(h_1^*, h_2^*)$ is dual to an edge (x_2, y_2) of h_2 . Since (we assume that) $VD^*(G, R)$ is a simple cycle separating G and R , it follows that in $VD^*(S)$, both x_1 and y_1 lie in green Voronoi cells (call these vertices *green* for short), and both x_2 and y_2 lie in red Voronoi cells (call them *red* vertices).

Since $VD^*(G, R)$ separates G and R (and since all dual vertices but h_1^* and h_2^* are of degree 3), the path $\pi^*(h_1^*, h_2^*)$ must contain an edge $e^* \in VD^*(G, R)$. Furthermore, we can partition $\pi^*(h_1^*, h_2^*)$ into three parts. The first part starts at the edge e_1^* dual to (x_1, y_1) and contains only edges whose duals (x, y) are such that both x and y are green. The last part ends at the edge e_2^*

dual to (x_2, y_2) and contains only edges whose duals (x, y) are such that both x and y are red. Between these green and the red parts of $\pi^*(h_1^*, h_2^*)$, there is a non-empty middle “red-green” part that contains edges whose duals (x, y) are such that x is green and y is red or vice versa. Our goal is to find an edge e^* in the middle part.

We find e^* by a binary search along $\pi^*(h_1^*, h_2^*)$, as follows. We locate the middle edge e_m^* of $\pi^*(h_1^*, h_2^*)$. Let (x, y) be the primal edge dual to e_m^* . We compute the closest site in S to x , and the closest site in S to y (We simply find, in “brute force”, the minimum among the distances from x to all the sites in S , and similarly for y .) If we discover that e^* is bichromatic (in the sense just mentioned) then we are done, and e^* is an edge that we seek. Otherwise, if e^* is fully green (that is, both x and y are green), we continue the search in the part of $\pi^*(h_1^*, h_2^*)$ between e^* and e_2^* , and if e^* is fully red (both x and y are red), we continue the search in the part between e_1^* and e^* . This binary search take $O(b \log r) = \tilde{O}(b)$ time.

Case 2: The hole h_1 is contained in a single cell of $VD^*(G)$ and the hole h_2 is contained in a single cell of $VD^*(R)$ (that is, no edge of $VD^*(G, R)$ crosses any of $\partial h_1, \partial h_2$). Let g be the green site such that all the vertices of ∂h_2 lie in $\text{Vor}(g)$ in $VD(G)$, and let r be the red site such that all the vertices of ∂h_1 lie in $\text{Vor}(r)$ in $VD(R)$. We take the bisector $\beta^*(g, r)$ and intersect it (i) with the (boundary of the) cell of r in $VD^*(R)$, and (ii) with the (boundary of the) cell of g in $VD^*(G)$, using the procedure from Section 4.

If no intersection was found in both steps, the boundary of $\text{Vor}(r)$ in $VD(R)$ is disjoint from the boundary of $\text{Vor}(g)$ in $VD(G)$, and the bisector $\beta^*(r, g)$ separates these two boundaries. In this case $VD^*(G, R)$ is the bisector $\beta^*(r, g)$, and we are done.

Otherwise, assume, without loss of generality, that we have found an intersection in step (i). Let x^* be one of the Voronoi vertices returned by the procedure; x^* is a Voronoi vertex in $VD^*(R \cup \{g\})$, defined by r, g , and some other red site $r' \neq r$. Let x_1, x_2 , and x_3 be the primal vertices on the boundary of the face associated with x^* such that x_1 is closest to r , x_2 is closest to r' , and x_3 is closest to g . If x_1 is also closer to r than to any site in $G \setminus \{g\}$ then the edge e^* dual to the edge (x_1, x_3) is in $VD^*(G, R)$. If x_2 is closer to r' than to any site in $G \setminus \{g\}$ then the edge e^* dual to the edge (x_2, x_3) is in $VD^*(G, R)$. Finally, if both x_1 and x_2 are closer to some green site than to all other red sites then we know that $VD^*(G, R)$ must cross the boundary of $\text{Vor}^*(r)$ in $VD^*(R)$ in any of its two segments between x^* and h_2^* . (Note that the boundary of $\text{Vor}^*(r)$ in $VD^*(R)$ is a simple cycle that contains h_2^* .) Let $\pi^*(x^*, h_2^*)$ a segment of $\text{Vor}^*(r)$ between x^* and h_2^* . We find e^* by a binary search along this path, in a manner analogous to the binary search along $\pi^*(h_1^*, h_2^*)$ that we performed in Case 1 above.

Once we have computed $VD^*(G, R)$, we paste it with the appropriate portions of $VD^*(G)$ and $VD^*(R)$, to obtain $VD^*(S)$, in a straightforward manner as we combined $VD^*(Y, O)$ with $VD^*(Y)$ and $VD^*(O)$ in Section 6.1.

6.3 Three holes

The next stage is to compute all the “trichromatic” Voronoi diagrams, where each such diagram is of the form $VD^*(S_i \cup S_j \cup S_k)$, for distinct indices $1 \leq i < j < k \leq t$, where S_i (resp., S_j, S_k) consists of all the sites on the boundary of i -th hole (resp., j -th hole, k -th hole) of P .

Extending Lemma 2.3, we get the following interesting property.

Lemma 6.2. *Let S, S', S'' be three subsets of sites, each appears consecutively on the boundary of a single hole of P (when the holes are not distinct, the corresponding subsets are assumed to be separated along the common hole boundary). Then there are at most two trichromatic Voronoi vertices in $VD^*(S \cup S' \cup S'')$. That is, there are at most two faces f of P for which there exist sites*

$u \in S$, $v \in S'$, and $w \in S''$, such that each of the cells $\text{Vor}(u)$, $\text{Vor}(v)$, $\text{Vor}(w)$ contains a single vertex of f .

Proof. The proof proceeds essentially as in the proof of Lemma 2.3, by showing that the existence of three trichromatic vertices would lead to an impossible plane drawing of $K_{3,3}$. Here one set of vertices are points inside the faces dual to the vertices, and another set are points inside the holes whose boundaries contain the sets S , S' , S'' . (When the holes are not distinct, we draw a separate point inside the hole for each subset along its boundary; the fact that these subsets are separated allows us to embed the relevant edges in a crossing-free manner.) \square

Fix three distinct holes, call them h_1 , h_2 , h_3 , and let S_1 , S_2 , S_3 be the respective subsets of sites, such that S_i lies on ∂h_i , for $i = 1, 2, 3$. The preceding procedure has already computed the three bichromatic diagrams $VD^*(S_1 \cup S_2)$, $VD^*(S_1 \cup S_3)$, and $VD^*(S_2 \cup S_3)$. We now want to merge them into the trichromatic diagram $VD^*(S_1 \cup S_2 \cup S_3)$.

We take each bichromatic bisector of the form $\beta^*(u, v)$, for $u \in S_1$ and $v \in S_2$, that contains a Voronoi edge in $VD^*(S_1 \cup S_2)$, and note that there are only $O(|S_1| + |S_2|) = O(b)$ such bisectors. For each such bisector $\beta^*(u, v)$, we take the Voronoi cell $\text{Vor}^*(u)$ in $VD^*(S_1 \cup S_3)$ (assuming it is not empty), denote it for specificity as $\text{Vor}_{13}^*(u)$, collect all its boundary Voronoi edges that are segments of bisectors of the form $\beta^*(u, w)$, for $w \in S_3$, and intersect each such $\beta^*(u, w)$ with $\beta^*(u, v)$. That is, we apply the procedure of Section 4 to find the (zero or two) Voronoi vertices of $VD^*({u, v, w})$. If we get two vertices, we test each of them whether it lies on the Voronoi edge(s) contained in $\beta^*(u, w)$. If so, it is a Voronoi vertex of the trichromatic diagram, as is easily verified, and otherwise it is not. We keep applying this step for u fixed. When we are done, one of two cases can arise.

(i) No trichromatic vertices have been found along $\partial \text{Vor}_{13}^*(u)$. In this case, either the final cell $\text{Vor}^*(u)$ (in $VD^*(S_1 \cup S_2 \cup S_3)$) is equal to $\text{Vor}_{13}^*(u)$, or the boundary of the final cell is fully contained in the interior of $\text{Vor}_{13}^*(u)$. To determine which of these situations applies, we pick a single arbitrary vertex f^* of $\partial \text{Vor}_{13}^*(u)$, iterate over all the (S_1, S_2) -bisectors $\beta^*(u, v)$ (that involve u), and test whether the corresponding primal face f has a vertex nearer to v . If this happens for at least one v , the boundary of the final cell is fully contained in the interior of $\text{Vor}_{13}^*(u)$, and otherwise the cell remains unchanged. Note that the overall number of tests that we perform, over all $u \in S_1$, is proportional to $O(|S_1| + |S_2|) = O(b)$, so this step takes $\tilde{O}(b)$ time.

(ii) We find at least one (real) trichromatic vertex along $\partial \text{Vor}_{13}^*(u)$. (Recall that this can happen at most twice, over all nodes u , by Lemma 6.2.) The boundary of the final cell $\text{Vor}^*(u)$ then consists of paths and cycles, each of which either is fully contained in the cell boundary in $VD^*(S_1 \cup S_2)$, or is fully contained in the cell boundary in $VD^*(S_1 \cup S_3)$, or consists of pieces of both boundaries, glued together at the common trichromatic vertex or vertices. As in Case (i), and with the same time bound, we can identify portions of the first two kinds. Portions of the third kind can be obtained by tracing the boundaries of both cells from the common trichromatic vertex or vertices.

Repeating this step to each $u \in S_1$, we obtain all the final cells of the sites of S_1 . Fully symmetric procedures compute the final cells of the sites in S_2 and in S_3 .

This completes the description of the construction of $VD(S_1 \cup S_2 \cup S_3)$ for the sites on a fixed triple of holes. We repeat this for all (the constant number of) such triples, and obtain all the trichromatic diagrams, as desired.

As follows from the analysis presented above, the overall running time of the constructions is $\tilde{O}(b)$.

6.4 Handling multiple holes

The procedures presented so far construct all the trichromatic Voronoi diagrams, namely all the diagrams of the form $VD^*(S_i \cup S_j \cup S_k)$, for distinct indices $1 \leq i < j < k \leq t$, where S_i (resp., S_j , S_k) consists of all the sites on the boundary of i -th hole (resp., j -th hole, k -th hole) of P . The final step in the construction, presented in this subsection, produces the overall diagram, of the entire set S , from these partial diagrams. (We assume here that there are at least three holes, otherwise this stage is not needed.)

Clearly, every Voronoi vertex in $VD^*(S)$ is also a Voronoi vertex in some trichromatic diagram, so the set Q^* of all the trichromatic vertices, over all the $\binom{t}{3}$ triples of holes, is a superset of the actual vertices of $VD^*(S)$. (To be precise, these sets also include “monochromatic” and “bichromatic” vertices, determined by sites that belong to just one hole or to two holes.) To find the real final Voronoi vertices and edges, we proceed as follows.

For each bisector $\beta^*(u, v)$, say, with $u \in S_1$ and $v \in S_2$, where S_1 and S_2 are the subsets of sites on the boundaries of two fixed respective holes h_1, h_2 , we mark along $\beta^*(u, v)$ all the Voronoi edges and vertices that show up in any of the trichromatic diagrams $VD^*(S_1 \cup S_2 \cup S_k)$, for $k = 3, \dots, t$. We sort the resulting vertices in their order along $\beta^*(u, v)$, and thereby obtain a partition of $\beta^*(u, v)$ into “atomic” intervals, each delimited by two consecutive vertices.

We then count, for each atomic interval I , in how many trichromatic diagrams it is contained in a Voronoi edge (of that diagram). This is easy to do by computing this number for some initial interval in brute force (which takes $\tilde{O}(1)$ time), and then by updating the count by ± 1 as we move from one atomic interval to the next one.

As is clear from the construction, the portions of $\beta^*(u, v)$ that are Voronoi edges in the full diagram $VD^*(S)$ are precisely those that are covered by exactly $t - 2$ trichromatic Voronoi edges if $i \neq j$, or by $t - 1$ edges if $i = j$. (This is the number of possible indices $k \neq i, j$ in each of these cases.)

We can therefore identify in this manner all the true Voronoi edges along each bisector, and also the true Voronoi vertices (which are the endpoints of the true edges). Each true vertex arises in this manner three times. By matching these three occurrences, we can “stitch” together the edges and vertices of $VD^*(S)$ into a single planar graph that represents the diagram. It is also easy to augment this graph in situations where the diagram is not connected, using standard features of the DCEL data structure [12].

7 Preprocessing for sum / max queries

In this section we describe how to preprocess P in $\tilde{O}(r^2)$ time so that, given the cell $\text{Vor}^*(v)$ of a site v , we can return the vertex $w \in \text{Vor}(v)$ maximizing $d(u, w)$ in time that is nearly linear in the number of edges of $\text{Vor}^*(v)$. Let T be the shortest path tree rooted at v in P and let T^* be the cotree of T . (That is, T^* is the tree whose edges are the duals of the edges which are not in T .) Let h_∞ be the hole such that $v \in \partial h_\infty$ and let h_∞^* be the vertex dual to h_∞ . We root T^* at h_∞^* and when we refer to an edge $f^*g^* \in T^*$ then f^* is the parent of g^* in T^* .

We label the vertices of P according to their distance (in P) from v and we label each face f (and its corresponding dual vertex f^*) that is not a hole of P with the maximum label of a vertex incident to f . Recall that $\text{Vor}(v)$ is a connected subtree of T (Lemma 2.1), and $T \setminus \text{Vor}(v)$ is a forest. Therefore, by the duality of cuts and cycles in planar graphs, $\text{Vor}^*(v)$ is a set \mathcal{C}^* of non self-crossing vertex disjoint cycles in P^* (see [49]). The set \mathcal{C}^* consists of at most t cycles since each cycle $C^* \in \mathcal{C}^*$ either encloses a hole h or passes through a vertex dual to a hole h . See Figures 8, 9. We assume that \mathcal{C}^* contains a cycle C_0^* through h_∞^* that encloses all the other cycles in \mathcal{C}^* . (If

there is no such cycle then we add a dummy cycle containing all other cycles which is a self loop through h_∞^* .) Furthermore, since $\text{Vor}(v)$ is connected, no cycle $C_1^* \in \mathcal{C}^* \setminus \{C_0^*\}$ is nested within a cycle $C_2^* \in \mathcal{C}^* \setminus \{C_0^*\}$.

Our goal is to report the maximum label of a vertex that is enclosed by C_0^* and not enclosed by any $C^* \in \mathcal{C}^* \setminus \{C_0^*\}$.

Let C^* be a cycle in \mathcal{C}^* . We say that an edge f^*g^* of T^* *penetrates* $\text{Vor}^*(v)$ at C^* (or, in short, *penetrates* at C^*) if $f^* \in C^*$ and $g^* \in \text{Vor}^*(v)$ (g^* can be on $\partial\text{Vor}^*(v)$) and we say that an edge f^*g^* of T^* *exits* $\text{Vor}^*(v)$ at C^* (or, in short, *exits* at C^*) if $f^* \in \text{Vor}^*(v)$ (f^* can be on $\partial\text{Vor}^*(v)$) and $g^* \in C^*$.

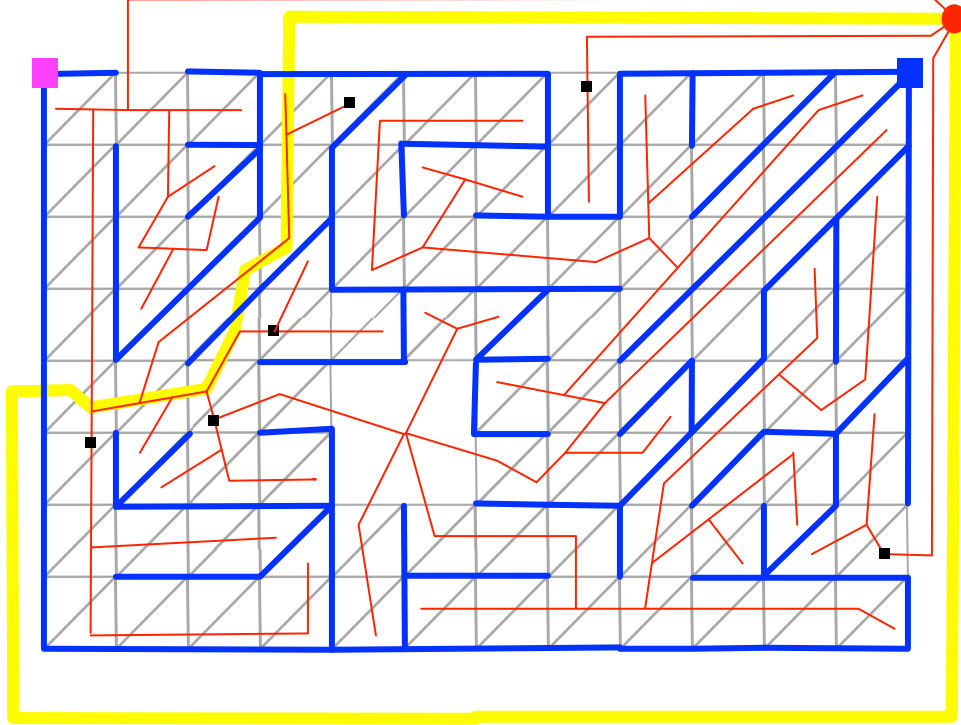


Figure 8: A graph with two holes. Two sites are shown, both on the external hole. The site on the upper right corner is v . In this example, $\text{Vor}^*(v)$ (yellow) consists of one simple cycle that goes through the infinite face. T is shown in blue and T^* in red. Endpoints of edges penetrating at C_0^* are indicated by black squares.

Lemma 7.1. *Every root-to-leaf path in T^* contains at most one edge penetrating at C_0^* and no edge exiting at C_0^**

Proof. Clearly, for a root-to-leaf path of T^* to contain two edges penetrating at C_0^* , there must be an edge between them on the path that is exiting at C_0^* . It therefore suffices to show there are no exiting edges. Since the root h_∞^* is not strictly enclosed by C_0^* , any root-to-leaf path of T^* must penetrate at C_0^* before exiting at C_0 . Suppose for the sake of contradiction that $f_0^*g_0^*$ is a penetrating edge and $f_1^*g_1^*$ is its descendant exiting edge. Consider the cycle formed by the f_0^* -to- g_1^* path in T^* and a f_0^* -to- g_1^* subpath of C_0^* . This cycle partitions $\text{Vor}(v)$ into two nonempty regions which gives a contradiction since T cannot cross this cycle. \square

To simplify the presentation we first assume that h_∞ is the only hole in P . We will remove this assumption later. If h_∞ is the only hole in P then C_0^* must be the only cycle in \mathcal{C}^* . Each vertex $u \in \text{Vor}(v)$ is either adjacent to an edge of C_0^* or to a face f which is strictly enclosed by C_0^* . By Lemma 7.1, for any penetrating edge $f^*g^* \in T^*$, the subtree of T^* rooted at g^* consists only of faces strictly enclosed by C_0^* . Conversely, since the root of T^* is not strictly enclosed by C_0^* , every face which is strictly enclosed by C_0^* belongs to the subtree of g^* for some penetrating edge $f^*g^* \in T^*$. It follows that it suffices to find the maximum label among the labels of the (primal) vertices of $\text{Vor}(v)$ adjacent to edges of C_0^* and of the labels of the dual vertices in each of the subtrees of T^* rooted at an edge penetrating at C_0^* .

Preprocessing. For each node $g^* \in T^*$ we compute the maximum label of a node in g^* 's subtree in T^* (denoted by $T_{g^*}^*$). Then we extend the computation of the binary search trees representing the bisectors $\beta^*(v, w)$ (see Section 3) to store with each dual vertex f^* on a bisector $\beta^*(v, w)$ a value $\ell(f^*)$, which is equal to the maximum label of a node in $T_{g^*}^*$, in case the edge $f^*g^* \in T^*$ penetrates $\beta^*(v, w)$, and 0 otherwise. With each edge e^* of $\beta^*(v, w)$ we store a value $\ell(e^*)$ which is equal to the label of the primal vertex closer to v that is adjacent to e^* . In addition, we store at the nodes of the binary search tree representing each bisector $\beta^*(v, w)$, the maximum of the $\ell()$ values of the edges and vertices in its subtree. This allows us to compute the maximum value of a node in a segment of $\beta^*(v, w)$ in logarithmic time. In addition, we construct a range maxima data structure (RMQ) for the edges incident to h_∞^* in their cyclic order around h_∞ , where the value $\ell(h_\infty^*g^*)$ of each edge $h_\infty^*g^*$ is defined to be the maximum label of a vertex of $T_{g^*}^*$.

Query. For the query, we are given the representation of C_0^* as a sequence of segments of bisectors $\beta^*(v, \cdot)$, and we need to report the maximum label of a vertex in $\text{Vor}(v)$. We query each segment γ^* of a bisector on C_0^* for the maximum value of $\ell(f^*)$ and $\ell(e^*)$ among the interior vertices f^* and the edges of γ^* , respectively. This takes $O(\log r)$ time per segment. In addition, for each two consecutive segments with a shared dual vertex f^* , we check whether the edge f^*g^* incident to f^* but not on C_0^* belongs to T^* and penetrates at C_0^* . If so, we take the maximum label of a node in $T_{g^*}^*$ as a candidate value for the maximum distance as well. Finally, if $g^*h_\infty^*$ and $h_\infty^*g'^*$ are two consecutive edges of C_0^* incident to h_∞^* , we query the RMQ data structure of h_∞^* for the maximum value associated with the edges (strictly) between $h_\infty^*g^*$ and $h_\infty^*g'^*$ in the cyclic order around h_∞^* that are enclosed by C_0^* . The desired maximum value is the largest among all these candidate values.

We now remove the assumption that there are no holes in P other than h_∞ . The challenge in this case is that a branch of T^* that penetrates C_0^* may go in and out of $\text{Vor}^*(v)$ through other cycles in \mathcal{C}^* . We show, however, that because the sites lie on a constant number of holes, there is a constant number of problematic branches of T^* and we can handle them with a special data structure. Recall that we would like to report the maximum label of a (primal) vertex strictly enclosed by C_0^* but not enclosed by any $C^* \in \mathcal{C}^* \setminus \{C_0^*\}$.

Consider traversing a root-to-leaf path Q in T^* . It starts with a prefix of faces that are not strictly enclosed by C_0^* , followed by an edge f^*g^* that penetrates $\text{Vor}^*(v)$ at C_0^* . By Lemma 7.1, the suffix of Q starting at g^* is strictly enclosed by C_0^* . This suffix of Q is not enclosed by any other cycle in \mathcal{C}^* until it uses an edge that exits $\text{Vor}^*(v)$ at some $C^* \in \mathcal{C}^*$. From that point on it is enclosed by C^* until it encounters an edge that penetrates at C^* , and so on. Therefore, we would like to report the maximum in any subtree rooted at g^* for any penetrating edge f^*g^* (at any $C^* \in \mathcal{C}^*$), but remove from each such subtree the subtrees rooted at g^* for any exiting edge f^*g^* . The next two lemmas characterize the exiting edges.

Lemma 7.2. *For every cycle $C^* \in \mathcal{C} \setminus C_0^*$ there is exactly one edge of T^* that exits at C^* .*

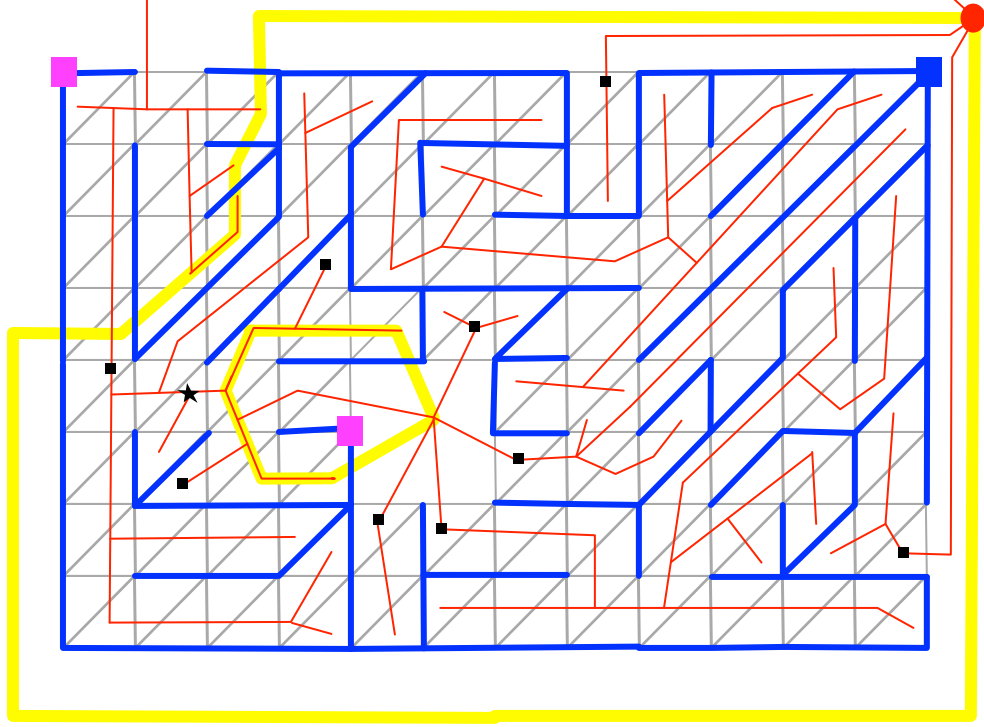


Figure 9: A graph with two holes. Three sites are shown, two on the external hole and one on the inner hole. The site on the upper right corner is v . In this example, $\text{Vor}^*(v)$ (yellow) consists two cycles. T is shown in blue and T^* in red. Endpoints of edges penetrating are indicated by black squares. An endpoint of an edge exiting $\text{Vor}^*(v)$ is indicated by a black asterix.

Proof. Since T^* is a spanning tree there is at least one such edge. Assume there are two edges $f_0^*g_0^*, f_1^*g_1^*$ exiting at C^* . The cycle formed by the portion of C^* between g_0^*, g_1^* and the unique g_0^* -to- g_1^* path in T^* (not necessarily directed) partitions $\text{Vor}(v)$ into two nonempty regions. A contradiction arises since T cannot cross this cycle. \square

Corollary 7.3. *Let $h = h_\infty$ be a hole and suppose that there is a cycle $C^* \in \mathcal{C}^*$ that either encloses h^* or goes through h^* . Then the unique edge of T^* that exits at C^* is on the path from h_∞^* to h^* in T^* .*

Proof. If the edge $g^*h^* \in T^*$ exits at C^* we are done. Otherwise, g^*h^* is enclosed by C^* (perhaps not strictly enclosed). Since h_∞^* is not enclosed by C^* , the path from h_∞^* to h^* in T^* must contain an edge that exits C^* . \square

Preprocessing (with multiple holes). For every hole $h \in \mathcal{H}$, let $T^*[h^*]$ denote the path from h_∞^* to h^* (the vertex dual to H) in T^* . Consider the subtree $T_{\mathcal{H}}^*$ of T^* which is the union of the paths $T^*[h^*]$ over all holes $h \in \mathcal{H}$.

We construct a data structure that can answer the following queries on $T_{\mathcal{H}}^*$. Given dual vertices $v_1^*, v_2^*, \dots, v_k^*$ in $T_{\mathcal{H}}^*$ (where $k \leq t$) such that v_1^* is an ancestor of all the other v_i^* 's, return the maximum label of a vertex in $T_{v_1^*}^* \setminus \cup_i T_{v_i^*}^*$. We obtain this data structure by contracting all the edges of $T^* \setminus T_{\mathcal{H}}^*$ into their ancestors in $T_{\mathcal{H}}^*$, keeping in each vertex v^* of $T_{\mathcal{H}}^*$ the maximum label of a vertex in the subgraph contracted to v^* . We then represent $T_{\mathcal{H}}^*$ (with these maxima as the vertex

values) by an Euler tour tree [35]. We can construct this data structure within the $\tilde{O}(r)$ time that it takes to compute T^* .

At the vertices f^* of the bisectors $\beta^*(v, w)$ we store the labels $\ell(f^*)$ which are defined as before with the difference that we set $\ell(f^*) = 0$ if the penetrating edge f^*g^* is in $T_{\mathcal{H}}^*$ (these edges will be taken into account separately). We also store, for each vertex $f^* \in \beta^*(v, w)$, which does not correspond to a hole, a boolean flag $b(f^*)$ which is 1, if the edge incident to f^* that is not an edge of $\beta^*(v, w)$, belongs to $T_{\mathcal{H}}^*$. We maintain these flags so that we can list all such dual vertices in any segment of $\beta^*(v, w)$ in $O(\log r)$ per dual vertex.

Finally, we maintain an RMQ data structure over the edges of each hole h in their cyclic order around h . The value $\ell(h^*g^*)$ of each edge h^*g^* is defined to be the maximum label of a vertex in T_g^* if $h^*g^* \in T^* \setminus T_{\mathcal{H}}^*$ and 0 otherwise. We also store with h^* the indices of the (constant number of) edges of $T_{\mathcal{H}}^*$ that are incident to h^* .

Query (with multiple holes). We traverse the cycles $C^* \in \mathcal{C}^*$. For each cycle C^* , we query the representation of each segment γ^* of a bisector $\beta^*(v, \cdot)$ in C^* and find the maximum value of $\ell(f^*)$ over all interior vertices $f^* \in \gamma^*$ and edges $e^* \in \gamma^*$. This takes $O(\log r)$ time per segment. The maximum in each segment is a candidate for the maximum distance from v to a vertex in $\text{Vor}(v)$. In addition, for each pair of consecutive segments with a common dual vertex f^* , we check whether the third edge f^*g^* belongs to $T^* \setminus T_{\mathcal{H}}^*$ and penetrates at C^* . If so, we consider the maximum label in T_g^* as a candidate for the maximum as well. Finally, for each hole h that C^* crosses we identify the pair of consecutive edges g^*h^* and $h^*g'^*$ of C^* which are incident to h^* , and we query the RMQ of h^* for the maximum value associated with the edges which are between g^*h^* and $h^*g'^*$ in the cyclic order of the edges around h^* and inside $\text{Vor}^*(v)$. We pick the largest among all these candidates as a candidate for the maximum distance as well.

It remains to handle vertices of \mathcal{C}^* that belong to $T_{\mathcal{H}}^*$. For this we collect a set S of edges while traversing the cycles in \mathcal{C}^* as follows. For each dual vertex f^* such that the flag $b(f)$ is 1 we put the edge f^*g^* incident to f^* which is not in C^* in S . From the definition of the flags, the edges in S are in $T_{\mathcal{H}}^*$. Note that each edge $f^*g^* \in S$ either penetrates or exits a cycle $C^* \in \mathcal{C}^*$, and furthermore there can be multiple penetrating edges collected at the same cycle C^* . For each hole h such that h^* is in some cycle C^* we add to S the edges of $T_{\mathcal{H}}^*$ incident to h^* (and in $\text{Vor}(v)$).

If we consider all the edges in S on a root-to-leaf path in $T_{\mathcal{H}}^*$, they must alternate between penetrating and exiting (and the first such edge surely penetrates). Since there are $O(1)$ leaves in $T_{\mathcal{H}}^*$ and also $O(1)$ collected edges that exit, we have that $|S| = O(1)$. We then use the Euler tour tree data structure for $T_{\mathcal{H}}^*$ to retrieve the contribution of each subtree T^* rooted at the vertex internal to $\text{Vor}^*(v)$ of each penetrating edge of S . Let v_1^* be an endpoint, internal to $\text{Vor}^*(v)$, of a penetrating edge in S . We identify all the exiting edges in S which are descendants of v_1^* . Let $v_2^*, \dots, v_j^* \in S$ be the endpoints, internal to $\text{Vor}^*(v)$, of these exiting edges. We query the Euler tour data structure for $T_{\mathcal{H}}^*$ with $v_1^*, v_2^*, \dots, v_j^*$ and take the returned value as another candidate for the maximum distance. Specifically, we identify the queries to the Euler tour data structure for $T_{\mathcal{H}}^*$ as follows. Let S' be the set of endpoints, internal to $\text{Vor}^*(v)$, of the edges in S . We define the depth of a vertex $g^* \in S'$ to be the number of vertices in S' that are proper ancestors of g^* in $T_{\mathcal{H}}^*$. For every i and for every $g^* \in S$ of depth $2i$, we query the Euler tour data structure for $T_{\mathcal{H}}^*$ with $v_1^* = g^*$, and $v_2^*, v_3^*, \dots, v_j^*$ the descendants of g^* in $T_{\mathcal{H}}^*$ that belong to S' and have depth $2i + 1$. Since $|S| = O(1)$, we perform a constant number of queries, and, can compute the appropriate vertices for each query in $O(1)$ time.

References

- [1] A. Abboud and S. Dahlgaard. Popular conjectures as a barrier for dynamic planar graph algorithms. In *FOCS*, pages 476–486, 2016.
- [2] A. Abboud, F. Grandoni, and V. V. Williams. Subcubic equivalences between graph centrality problems, APSP and diameter. In *SODA*, pages 1681–1697, 2015.
- [3] A. Abboud and K. Lewi. Exact weight subgraphs and the k -SUM conjecture. In *ICALP*, pages 1–12, 2013.
- [4] A. Abboud and V. V. Williams. Popular conjectures imply strong lower bounds for dynamic problems. In *FOCS*, pages 434–443, 2014.
- [5] A. Abboud, V. V. Williams, and J. R. Wang. Approximation and fixed parameter subquadratic algorithms for radius and diameter in sparse graphs. In *SODA*, pages 377–391, 2016.
- [6] A. Abboud, V. V. Williams, and H. Yu. Matching triangles and basing hardness on an extremely popular conjecture. In *STOC*, pages 41–50, 2015.
- [7] A. Aggarwal, M. M. Klawe, S. Moran, P. Shor, and R. Wilber. Geometric applications of a matrix-searching algorithm. *Algorithmica*, 2(1):195–208, 1987.
- [8] F. Aurenhammer. Voronoi diagrams - A survey of a fundamental geometric data structure. *ACM Comput. Surv.*, 23(3):345–405, 1991.
- [9] A. Backurs, N. Dikkala, and C. Tzamos. Tight hardness results for maximum weight rectangles. In *ICALP*, pages 81:1–81:13, 2016.
- [10] A. Backurs and C. Tzamos. Improving Viterbi is hard: Better runtimes imply faster clique algorithms. *Arxiv 1607.04229*, 2016.
- [11] B. Ben-Moshe, B. K. Bhattacharya, Q. Shi, and A. Tamir. Efficient algorithms for center problems in cactus networks. *Theor. Comput. Sci.*, 378(3):237–252, 2007.
- [12] M. d. Berg, O. Cheong, M. v. Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 3rd edition edition, 2008.
- [13] S. Cabello. Subquadratic algorithms for the diameter and the sum of pairwise distances in planar graphs. In *SODA*, pages 2143–2152, 2017. Full version in *Arxiv 1702.07815*.
- [14] S. Cabello, E. Chambers, and J. Erickson. Multiple-source shortest paths in embedded graphs. *SIAM Journal on Computing*, 42(4):1542–1571, 2013.
- [15] T. M. Chan. All-pairs shortest paths with real weights in $O(n^3/\log n)$ time. *Algorithmica*, 50(2):236–243, 2008.
- [16] T. M. Chan. More algorithms for all-pairs shortest paths in weighted graphs. *SIAM J. Comput.*, 39(5):2075–2089, 2010.
- [17] V. Chepoi and F. F. Dragan. A linear-time algorithm for finding a central vertex of a chordal graph. In *ESA*, pages 159–170, 1994.

- [18] V. Chepoi, F. F. Dragan, B. Estellon, M. Habib, and Y. Vaxès. Diameters, centers, and approximating trees of delta-hyperbolic geodesic spaces and graphs. In *SoCG*, pages 59–68, 2008.
- [19] V. Chepoi, F. F. Dragan, and Y. Vaxès. Center and diameter problems in plane triangulations and quadrangulations. In *SODA*, pages 346–355, 2002.
- [20] K. Clarkson and P. Shor. Applications of random sampling in computational geometry, ii. *Discrete & Computational Geometry*, 4(5):387–421, 1989.
- [21] V. Cohen-Addad, S. Dahlgaard, and C. Wulff-Nilsen. Fast and compact exact distance oracle for planar graphs. *Arxiv 1702.03259*, 2017.
- [22] É. C. de Verdière. Shortest cut graph of a surface with prescribed vertex set. In *ESA*, pages 100–111.
- [23] W. Dobosiewicz. A more efficient algorithm for the min-plus multiplication. *International Journal of Computer Mathematics*, 32(1-2):49–60, 1990.
- [24] F. F. Dragan and F. Nicolai. Lexbfs-orderings of distance-hereditary graphs with application to the diametral pair problem. *Discrete Applied Mathematics*, 98(3):191–207, 2000.
- [25] D. Eppstein. Subgraph isomorphism in planar graphs and related problems. In *SODA*, pages 632–640, 1995.
- [26] J. Fakcharoenphol and S. Rao. Planar graphs, negative weight edges, shortest paths, and near linear time. *J. Comput. Syst. Sci.*, 72(5):868–889, 2006.
- [27] A. M. Farley and A. Proskurowski. Computation of the center and diameter of outerplanar graphs. *Discrete Applied Mathematics*, 2(3):185–191, 1980.
- [28] R. Floyd. Algorithm 97: Shortest path. *Commun. ACM*, 5(6):345, 1962.
- [29] G. N. Frederickson. Fast algorithms for shortest paths in planar graphs, with applications. *SIAM J. Comput.*, 16(6):1004–1022, 1987.
- [30] M. Fredman. New bounds on the complexity of the shortest path problem. *SIAM J. Comput.*, 5(1):83–89, 1976.
- [31] Y. Han. Improved algorithm for all pairs shortest paths. *Information Processing Letters*, 91(5):245–250, 2004.
- [32] Y. Han. An $O(n^3(\log \log n / \log n)^{5/4})$ time algorithm for all pairs shortest paths. *Algorithmica*, 51(4):428–434, 2008.
- [33] Y. Han and T. Takaoka. An $O(n^3 \log \log n / \log^2 n)$ time algorithm for all pairs shortest paths. In *SWAT*, pages 131–141, 2012.
- [34] D. Hartvigsen and R. Mardon. The all-pairs min cut problem and the minimum cycle basis problem on planar graphs. *Journal of Discrete Mathematics*, 7(3):403–418, 1994.
- [35] M. R. Henzinger and V. King. Maintaining minimum spanning trees in dynamic graphs. In *ICALP*, pages 594–604, 1997.

- [36] M. R. Henzinger, P. N. Klein, S. Rao, and S. Subramanian. Faster shortest-path algorithms for planar graphs. *Journal of Computer and System Sciences*, 55(1):3–23, 1997.
- [37] J. Hershberger and S. Suri. Matrix searching with the shortest-path metric. *SIAM J. Comput.*, 26(6):1612–1634, 1997.
- [38] T. Husfeldt. Computing graph distances parameterized by treewidth and diameter. In *IPEC*, pages 16:1–16:11, 2016.
- [39] P. Klein and S. Mozes. Optimization algorithms for planar graphs. <http://planarity.org>. Book draft.
- [40] P. N. Klein. Multiple-source shortest paths in planar graphs. In *16th SODA*, volume 5, pages 146–155, 2005.
- [41] P. N. Klein, S. Mozes, and C. Sommer. Structured recursive separator decompositions for planar graphs in linear time. In *STOC*, pages 505–514, 2013.
- [42] R. Klein. *Concrete and Abstract Voronoi Diagrams*, volume 400 of *Lecture Notes in Computer Science*. Springer, 1989.
- [43] R. Klein, E. Langetepe, and Z. Nilforoushan. Abstract voronoi diagrams revisited. *Comput. Geom.*, 42(9):885–902, 2009.
- [44] R. Klein, K. Mehlhorn, and S. Meiser. Randomized incremental construction of abstract voronoi diagrams. *Comput. Geom.*, 3:157–184, 1993.
- [45] D. Marx and M. Pilipczuk. Optimal parameterized algorithms for planar facility location problems using voronoi diagrams. In *ESA*, pages 865–877, 2015.
- [46] S. Olariu. A simple linear-time algorithm for computing the center of an interval graph. *International Journal of Computer Mathematics*, 34:121–128, 1990.
- [47] L. Roditty and V. V. Williams. Fast approximation algorithms for the diameter and radius of sparse graphs. In *STOC*, pages 515–524, 2013.
- [48] L. Roditty and U. Zwick. On dynamic shortest paths problems. *Algorithmica*, 61(2):389–401, 2011.
- [49] N. Sarnak and R. E. Tarjan. Planar point location using persistent search trees. *Commun. ACM*, 29(7):669–679, 1986.
- [50] M. I. Shamos and D. Hoey. Closest-point problems. In *SFCS*, pages 151–162, 1975.
- [51] T. Takaoka. A new upper bound on the complexity of the all pairs shortest path problem. *Information Processing Letters*, 43(4):195–199, 1992.
- [52] T. Takaoka. An $O(n^3 \log \log n / \log n)$ time algorithm for the all-pairs shortest path problem. *Information Processing Letters*, 96(5):155–161, 2005.
- [53] S. Warshall. A theorem on boolean matrices. *J. ACM*, 9(1):11–12, 1962.
- [54] R. Williams. Faster all-pairs shortest paths via circuit complexity. In *STOC*, pages 664–673, 2014.

- [55] V. V. Williams and R. Williams. Subcubic equivalences between path, matrix and triangle problems. In *FOCS*, pages 645–654, 2010.
- [56] V. V. Williams and R. Williams. Finding, minimizing, and counting weighted subgraphs. *SIAM J. Comput.*, 42(3):831–854, 2013.
- [57] C. Wulff-Nilsen. Wiener index and diameter of a planar graph in subquadratic time. Technical report, 08-16, Department of Computer Science, University of Copenhagen, 2008. <http://www.diku.dk/OLD/publikationer/tekniske.rapporter/rapporter/08-16.pdf>. Preliminary version in EurCG 2009.
- [58] C. Wulff-Nilsen. Wiener index, diameter, and stretch factor of a weighted planar graph in subquadratic time, 2010. Paper M in the PhD thesis of C. Wulff-Nilsen, available at <http://www.diku.dk/forskning/phd-studiet/phd/ThesisChristian.pdf>.
- [59] U. Zwick. A slightly improved sub-cubic algorithm for the all pairs shortest paths problem with real edge lengths. In *ISAAC*, pages 921–932, 2004.